4-1-1996

A Relationship in 2AFC experiment descovered using Monte Carlo Method

Dalei Huang

A RELATIONSHIP IN 2AFC EXPERIMENT DISCOVERED USING MONTE CARLO METHOD

by

Dalei Huang

B.S. Beijing University of Posts and Telecommunications

(1988)

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Sciences
in the Center for Imaging Science
of the Rochester Institute of Technology

April 1996

Signature of the Author_____

Accepted by_____ Oct. 3, 1996
Coordinator, M.S. Degree Program Date

CENTER FOR IMAGING SCIENCE
ROCHESTER INSTITUTE OF ROCHESTER
ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

M.S. DEGREE THESIS

The M.S. Degree Thesis of Dalei Huang
has been examined and approved by the
thesis committee as satisfactory for the
thesis requirement for the
Master of Science degree

Dr. Mark Fairchild, Thesis Advisor

Dr. Jeff Pelz

Dr. Ethan Montag

Oct, 3, 1996
Date

THESIS RELEASE PERMISSION
ROCHESTER INSTITUTE OF ROCHESTER
CENTER FOR IMAGING SCIENCE

Title of Thesis:

A Relationship in 2AFC Experiment Discovered Using Monte Carlo Method

I, Dalei Huang, hereby grant permission to the Wallace Memorial Library of R.I.T. to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Signature:_____

Date: Oct, 3, 1996

A RELATIONSHIP IN 2AFC EXPERIMENT DISCOVERED USING MONTE CARLO METHOD

by

Dalei Huang

Submitted to the
Center for Imaging Science
in partial fulfillment of the requirements
for the Master of Science Degree
at the Rochester Institute of Technology

Abstract

In detection theory, the Two Alternative Forced Choice (2AFC) experiment is becoming more preferable to experimenters, especially when synthetic images are used. A key measure of human performance in 2AFC experiments is the detectability index, d'. There are two different ways to calculate d'. The first is based on the average of correct responses, and the second is based on the average of z-scores for both signal sides. The two values are usually different. However the normalized difference of the two

values have a relationship with the estimated bias, c. The relationship is approximately parabolic, and can be expressed as $\Delta d' = a \cdot c^2$, where the coefficient, a , is nearly -0.5.

The main approach I have adopted in this thesis uses Monte Carlo simulation. Thus, most experiments are simulated on the computer. In this thesis, this method is described in detail. The programs used for the simulations are in the Appendix. The results obtained from this method are compared with the results from human observers and real experiments. The conclusion is that the results obtained from Monte Carlo experiments very closely reflects the results from the real experiments. The Monte Carlo simulation method very accurately simulates human performance.

Dedicated to my wife, Bin He,
and my whole family.

ACKNOWLEDGMENT

I would like to thank Dr. Mark Fairchild for his guidance and advice which made this work possible.

I would like to thank Dr. Jeff Pelz and Dr. Ethan Montag for their participation on my committee and for suggetions that made this thesis a reality.

I would like to thank Dr. Art Burgess for his help.

I would like to thank Dr. Dana Marsh for his kind help during my study at the center.

Contents

List of Figures

1. Introduction

For many years, detection theory has been applied successfully in medical image analysis. It provides a psychophysical approach to measure a viewer's performance distinguishing the weak signals from a "noisy" background.

Generally speaking, when a diagnostician tries to judge a medical image, he or she attempts to discern specific medical information from it. The relationship of the viewer's response to a stimuli provides an objective standard to evaluate a individual's performance. In such judgments, both correct and incorrect results can occur. There are two types of correct results: hits and correct rejects. Incorrect results can be classified into two other types: misses and false alarms. For instance, the diagnostician misses the shadow of a tumor, or reports the presence of the shadow which does not exist. Both of these errors generate a bad result. Errors are inevitable, and can arise for various reasons, such as weak signals

against a strong noise, or an observer's visual acuity limitation. The reasons which cause errors can not be removed completely. A diagnostician's performance can be improved by increasing the signal to noise ratio, or by a period of training. The detection theory is not intended to improve that ratio, but does present an effective way to measure human perception and accuracy.

There are two ways to assess human observer performance of image based decision tasks. The first is called the Receiver Operating Characteristic (ROC) method, the second is called the Multiple Alternative Forced Choice method. The former is more likely to be adopted when the limiting constraint is the number of images, and the latter is preferable while using synthetic images in the experiments.

This thesis mainly focuses on the Two Alternative Forced Choice (2AFC) method. Most of the 2AFC experiments in this thesis have used the Monte Carlo simulation method. The rest of the

experiments have used human observer method. There are two ways to calculate the detectability index, d'. The difference is that, one is to average in P (the probability) domain, and the other is to average in z (the z-score) domain. The first, assuming unbiased behavior, is to average the correct responses over both left and right sides, then measure the percentage of total correct responses, then transform this percentage to a z-score, and then transform the z-score to a dectectability index, d'. The second way is introduced by Macmillan and Creelman [1]. They measure the percentages of correct responses over left and right sides respectively, transform them to 2 z-scores, then average those z-scores, and transform that average to a detectability index, d'. To distinguish the 2 detectability indices calculated with the different methods, the first d' is marked as d'_p, and the second one is marked as d'_z. Swensson and Judy [9] calculated the difference between the two values, $\Delta d' = d'_p - d'_z$, and present a plot with $\Delta d'$ as vertical axis and the estimated bias, c, the tendency of an observer to give one response more than another, as

horizontal axis. The data used for plotting were from many observers and many experiments. Plotting the above data, we see an approximately inverted parabola that is symmetrical about the zero bias point. This result motivated us to do further research on the 2AFC experiments to determine the relationship between the two measures. The approach we chose is the Monte Carlo method to simulate the 2AFC experiments, so that the signal amplitude and the judgment threshold can be easily changed and controlled. This thesis will describe the design of our Monte Carlo experiments, the analysis of the experiments results, and the findings of the relationship of the d' difference and the estimated bias, c.

2. Theory

In the Receiver Operating Characteristic (ROC) method, the observer is presented with one image at a time. This image may or may not contain a signal. The observer has to decide if a signal is present. The observer is asked to respond "yes" when he or she considers the signal is shown, or "no" when he or she thinks the signal is not there. It turns out that the observer can make two kinds of correct responses and two types of errors. The two types of correct responses are correct positive (the observer says "yes" while the signal is there), and correct negative (the observer says "no" while no signal is shown). The two types of errors are false positive (the observer says "no" when the signal does exist), and false negative (the observer says "yes" and there is no signal). Each observer has his own unique decision criterion for judging a series of images. This person should maintain his criterion over this block of trials. The fractions of “true” and “false” positive responses can be obtained and thus generate a point on the ROC curve. As he varies the criterion for his decision, the probabilities of these two responses change

accordingly, and result in different points in the ROC curve. The ROC curve finally comes out as a plot of the covariation of the true positive response fraction as a function of the false positive response fraction. The following plot shows some ROC curves.

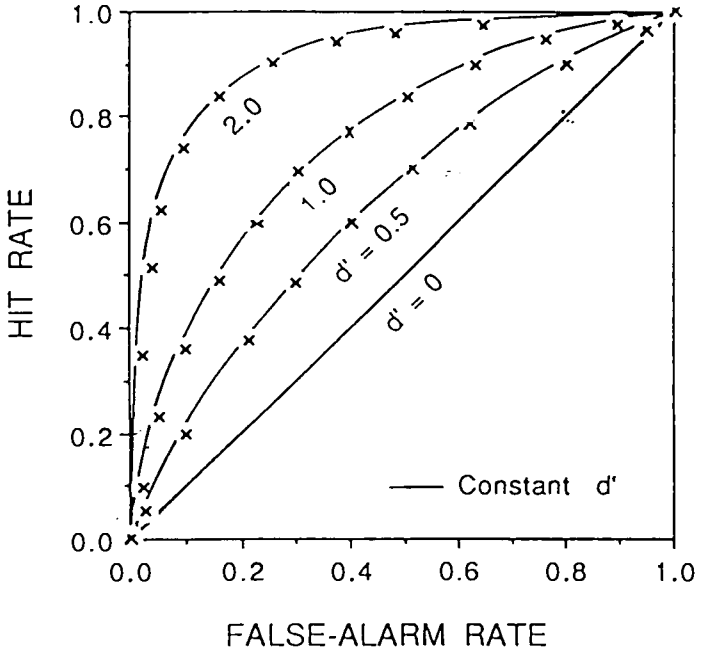


figure 2.1. ROC curves. The points on the same curves have constant d′ [1]

In order to estimate the performance of an observer, we need to make certain measurements. In the ROC experiment, the measures are A_z and d_a. A_z is the area under the ROC, while d_a, the

corresponding detectability index, is a mathematical transformation of A_z.

The theory behind the analysis of signal detection experiments is based on the assumption that every different judgment event involved in each task can be mapped onto a single dimension random variable, which is the decision variable. For a ROC experiment, the two hypotheses are signal sample <s> , and the noise sample <n>.

In the two alternative forced choice method (2AFC), the observer is presented with one image with two noise fields. The noise usually is uncorrelated random Gaussian noise. One of the two noise fields contains a signal with a specified shape, size and intensity. The other contains only noise. The probability of assigning the signal to either of the two fields is equal, but the observer doesn't know which field contains the signal. The observer is forced to make a decision about which field contains the signal. These responses are divided into four types, correct right, correct left, false right and false left.

In the 2AFC experiments, all results can be described by two measures. The proportion of correct response, P, and the detectability index, d'. The value of d' is also obtained as a transformation of P. The domains of the two measures are different. The value of P falls in the range of zero to one. However, since the completely correct judgment results in P equal to 1, and arbitrarily guessing results in P equal to 0.5, we are just concerned about the range from 0.5 to 1. And these two end points represent the worst and the best performance respectively. Although the value of d' falls within the range from minus infinity to plus infinity, it equals zero for P equal to 0.5, and equals infinity for P equal to 1. We are just concerned about the range from 0 to positive infinity, which corresponds to the worst and best observer's performance. The meaning of the detectability index d' can be simply understood as a number obtained by a defined transformation of P for 2AFC experiments. The larger the number, the better the performance of the observer. The minimum value for d' is 0. Therefore, the detectability index does reflect a feature of the observer's performance.

For the 2AFC experiment, the judgment event can also be mapped to a decision variable, since the signal can show up randomly in either the right or left interval. The two possible hypothesis can be <sn> and <ns>. The probability density function (PDF) and covariance function of the decision variable has to be predefined before we do further analysis of the detectability index, d'. Otherwise we can not figure out the d' value from the probability P. In practice, a number of assumptions are made about the PDFs to make the analysis of the signal detection experiments easy and convenient. There are some basic assumptions that are widely used, first the alternative intervals are statistically independent, second all PDFs have Gaussian distributions, and third all variances are the same. All the analysis to follow is based on these assumptions.

It is time to introduce the parameter z-score at this point. Given the above assumptions, the z-score is defined as the integral of the cumulative normal distribution with regard to the probability , P.

$$P(z) = \Phi(z) = \frac{1}{\sqrt{2\pi}} \cdot \int_{-\infty}^{z} exp(-x^2/2)dx \qquad (2.1)$$

the inverse relation is : $z(P) = \Phi^{-1}(P)$ (2.2)

This relationship is usually described in tabular form for convenience. A few examples of corresponding pairs are shown below. (P; z) can be any one of (0.2, -0.842), (0.3, -0.524), (0.5, 0.0), (0.6, 0.253), (0.9, 1.282), [3] etc.

A few other related terms also need to be defined here in order to describe the calculation method of detectability index. In the 2AFC experiment, the proportion of correct responses when the signal is on the left is defined as P_l, and the proportion of the correct response when the signal is on the right is defined as P_r. The proportion of incorrect responses while the signal being on right is defined as P_{r_e} , and the proportion of incorrect response while the signal being on left is defined as P_{l_e} .

Suppose the total number of trials in an experiment is N, the number of trials in which the left interval contains the signal is N_l, the number of trials in which the right interval contains the signal is N_r, then we have:

$$N_l + N_r = N. \tag{2.3}$$

Assuming in the N_l trials, the number of correct responses from the observer is N_{l_c}, the number of incorrect responses is N_{l_e}. The incorrect responses here mean that the observer selected the right interval in these trials.

$$N_{l\_c} + N_{l\_e} = N_l \tag{2.4}$$

$$P_l = N_{l\_c} / N_l \tag{2.5}$$

$$P_{r\_e} = N_{l\_e} / N_l \tag{2.6}$$

$$P_l = 1 - P_{r\_e} \tag{2.7}$$

Assuming in the N_r trials, the number of correct responses from the observer is N_{r_c}, the number of incorrect responses is N_{r_e}.

The incorrect responses here mean that the observer selected the left interval in these trials.

$$N_{r\_c} + N_{r\_e} = N_r \tag{2.8}$$

$$P_r = N_{r\_c} / N_r \tag{2.9}$$

$$P_{l\_e} = N_{r\_e} / N_r \tag{2.10}$$

$$P_r = 1 - P_{l\_e} \tag{2.11}$$

So the four parameters describing the 2AFC trials are not completely independent. Two parameters of the four are enough to describe the features in 2AFC experiment. The two chosen here are P_l (the proportion of the correct response when the signal is on the left) and P_r (the proportion of the correct response when the signal is on the right).

Then one can calculate the final observer detectability index. If the number of trials with the signal in left side equals the number with signal in right side, i.e. $N_l = N_r$, one can calculate d' using an unweighted average of z-scores,

$$d'_{zu} = \sqrt{2}\ [\ z(P_l) + z(P_r)]\ /\ 2 \qquad (2.12)$$

One can also calculate the unweighted average of probability of correct responses,

$$P_u = [\ P_r + P_l\ ]\ /2 \qquad (2.13)$$

and then derive

$$d'_{pu} = \sqrt{2}\ z(P_u) \qquad (2.14)$$

If N_l and N_r are not equal, which is the usual case, then it might be more reasonable to calculate the final observer detectability index by using a weighted z-score average. The weight for trials with signal on left should be $W_l = N_l/(N_l + N_r)$, and the weight for trials with signal on right be $W_r = N_r/(N_l + N_r)$. And the d' is

$$d'_{zw} = \sqrt{2}\ [W_l \bullet z(P_l) + W_r \bullet z(P_r)] \qquad (2.15)$$

One can also calculate the weighted average proportion of correct responses:

$$P_W = (N_{l\_c} + N_{r\_c}) / (N_l + N_r) \qquad (2.16)$$

where N_{l_c} and N_{r_c} are the number of correct responses on the left and right.

Then, the final d'_{pw} can be calculated :

$$d'_{pw} = \sqrt{2} \bullet z(pw) \qquad (2.17)$$

Generally speaking, if the probability of signal assignments to left and right intervals are not equal, the weighted and unweighted d' will probably be different. And since the P to z-score transformation is non-linear, the d' resulted after averaging of probability of correct responses will be different from d' results after averaging z-scores.

The other important measure is the bias, which shows a tendency in favor of one response over another. In the 2AFC experiments, it shows the observer's preference for choosing the

signal on one interval over the other interval that contains a signal. The response bias can estimate either the degree to which a "left" response is preferred, or the degree to which a "right" response is preferred. The equation to calculate the bias is:

$$c = -0.5 [z(P_H) + z(P_F)] \quad (2.18)$$

When the false alarm rate equals the miss rate, $z(F) = z(1-H) = -z(H)$, the bias equals zero. When false alarm rate exceeds the miss rate, the bias is less than 0; when the false alarm rate is less than the miss rate, the bias is larger than 0.

3. Monte Carlo Approach

The Monte Carlo method is used here to simulate 2AFC experiments. A method of approximately solving problems by the simulation of random quantities is defined as Monte Carlo method. Before the computer appeared, it was very difficult to apply this method, since the simulation of random quantities is a very laborious process. Now, as computers are widely available, it is, in principle, easy to do Monte Carlo simulations.

In order to use the Monte Carlo method, pseudo random noise must be generated properly. The book *Numerical Recipes in C* [2] provides a good way to generate random Gaussian numbers of different means and variances. In the computer simulation of 2AFC experiment, the Gaussian random noise is produced using the method suggested in "*Numerical Recipes in C*" [2]. A program to test how good this Gaussian generator was written and is included in Appendix B. The diagram of the test program is shown below:

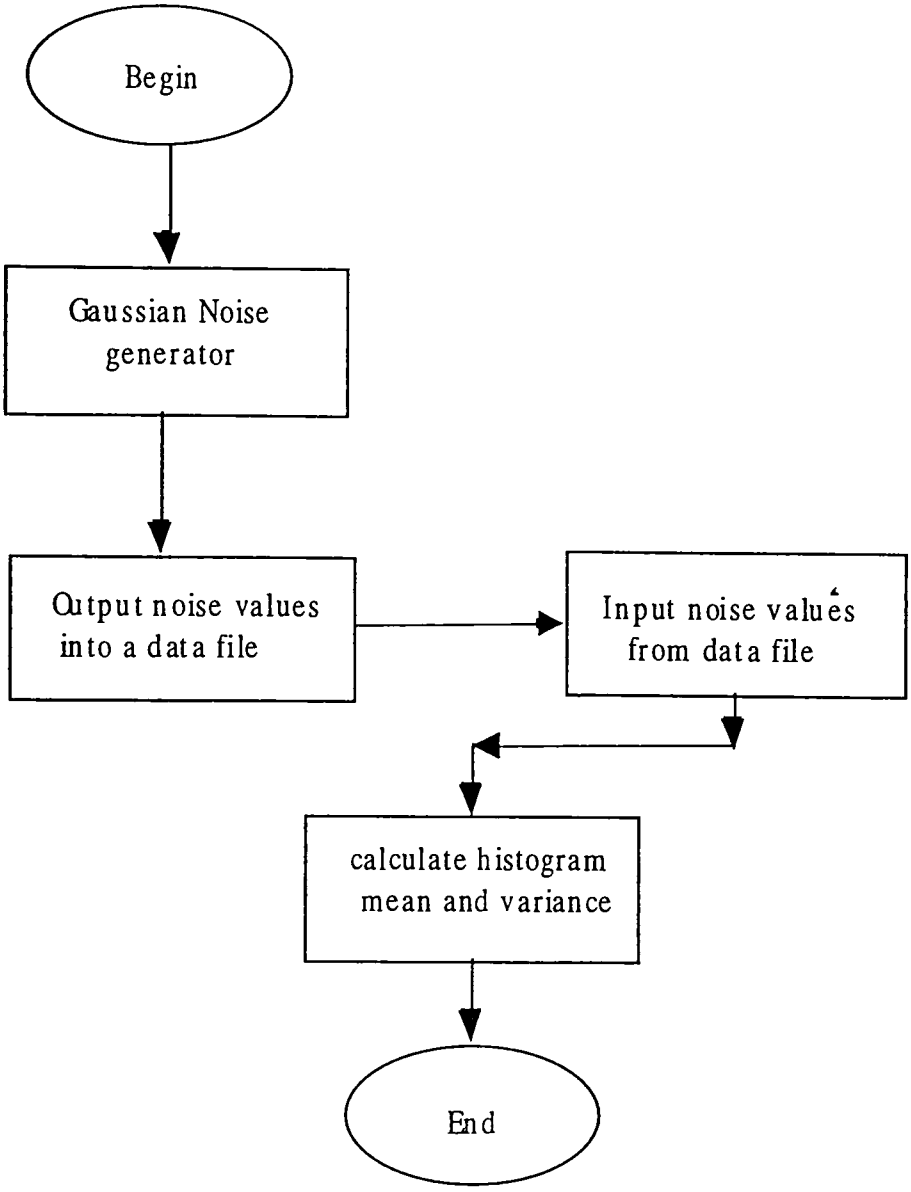


Figure 3.1. Diagram for calculating histogram. The c program is included in Appendix B.

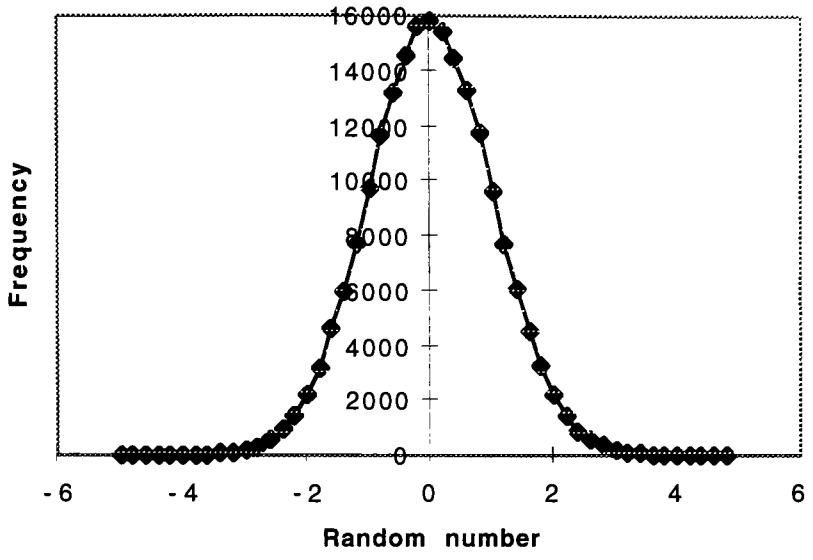
In order to test the Gaussian noise generator, I created two separate processes. The first is to generate random numbers, with parameters passed by tester. The parameters include mean, variance, and the number of random values wanted by the tester. The output data are directed to a data file. The second process is to read this data file, and calculate the histogram, mean and variance.

When testing this noise generator, 100,000 random numbers produced by this generator, with the parameters set to produce mean=0, variance=1.0, were sampled, and the following results were obtained.

The mean of the Gaussian numbers is 0.000703.

The variance of the Gaussian numbers is 1.000035.

The histogram of the random noise is shown below with the bin width equal to 0.2:



. Figure 3.2. The histogram of psudo-Gaussian noise. All the Gaussian noises used in this thesis were generated from this generator

We decided to use this generator throughout our experiments because the testing results of its performance were satisfactory. The C program of this generator is included in Appendix C.

The diagram of our Monte Carlo simulation experiment is shown on next page. The explanations follows the diagram.

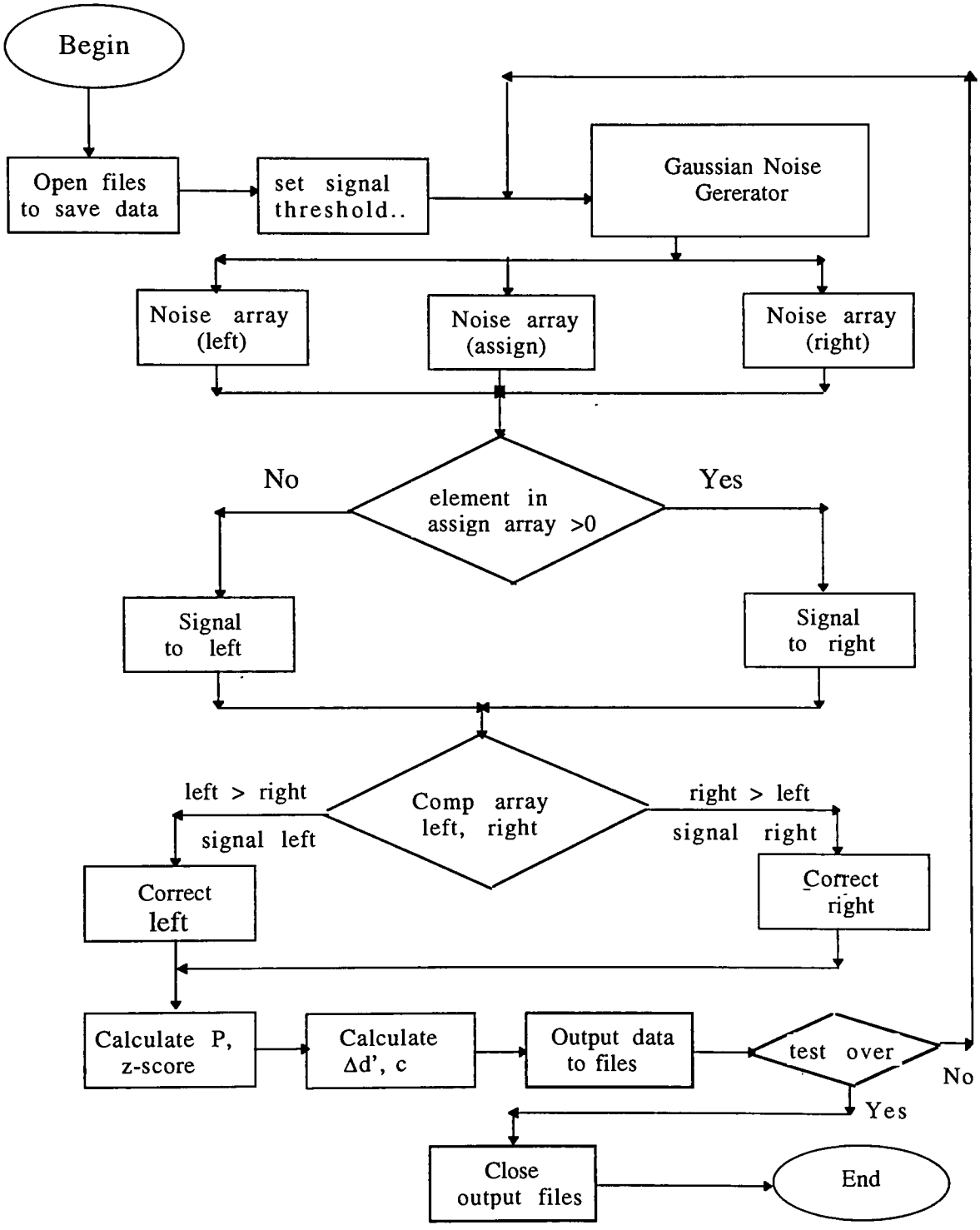


Figure 3.3. Diagram for computer simulation of Monte Carlo method

While generating the noise arrays, the seed has to be passed to the Gaussian noise generating routine as a parameter each time the routine is called. The seeds in this Monte Carlo experiment for generating noise arrays start from -509999, -507777, -505759 respectively. After one block has been tested, the seeds increase by a randomly selected number (5397) multiplied by the number of blocks which had already been tested. The seeds we chose were arbitrary. In fact, any number can be used here as a seed. The total blocks in any one experiment was fixed at one thousand.

In the simulation program, the routine is called three times for each test block. Each time the routine is called, an array of Gaussian numbers is generated. Because this simulates the 2AFC experiment, two arrays can be considered as two series of background images to be judged by the performers after the signal is added to one of them. The third way is used to assign the signal to one of the two sides randomly. If the noise value of the third array is greater than zero, the signal will be assigned to array number one, otherwise, the signal is assigned to array number two.

In the 2AFC experiment, the first array of random numbers is regarded to represent Gaussian noise on the left side, and is called left array. The second array of random numbers represents the Gaussian noise on the right side, and is called right array. The third array of numbers is used to assign signal, and is called the assignment array. The above three arrays have the same size in a test case. It changed from one group of test cases to another. We used 128, 256, 512, and 1024 as array size in different experiments. The value of the signal is set to one in half of all experiments, and set to two in the other half. When the signal carries value one, the signal to noise ratio (SNR) is equal to one. When the signal carries value two, the SNR is equal to two. Based on the sign of the number from the assignment array, the value of the signal is added to either right array or left array.

The simulator makes its judgment by comparing the values in the 2 arrays, which represent the left and right side images. If the value on the left side is larger than the one on the right side, it will

judge the signal is in the left array, otherwise, it will judge it in right array, no matter where the signal in fact is.

We, as the experiment designers, know where the signal is by the way we assign the signals. Because of the way of generating noise, the noise value can be very close to but never equal to 0. Thus we do not pay attention to the case of noise value equal to zero. We can classify the simulator's responses into 4 types, correct left, false left, correct right, and false right. The definition follows:

Correct left means the value of assignment is smaller than 0 (signal is assigned to left), while the value of the left array is greater than the one of the right array.

False right means the value of assignment is smaller than 0 (signal is assigned to left), while the value of the left array is smaller than the one of the right array.

Correct right means the value of assignment is greater than 0 (signal is assigned to right), while the value of the right array is greater than the one of the left array.

False left means the value of assignment is greater than 0 (signal is assigned to right), while the value of the right array is smaller than the one of the left array.

Thus we can calculate the probability of any type of the four.

From this probability, we can calculate the detectability d' and the bias c.

The following section will show some typical results from a large number of experiments.

4. Analysis and discussion

In order to make the test result as accurate as possible, we have done several hundred experiments. Thus, the data obtained from the experiments are sufficient for us to do analysis, although only a small part of the data were selected to be shown to demonstrate the results. The rest of the data say the same thing. One can get them by running the program appended in this thesis.

Before we analyze the experiment results, we need to define several concepts related to an experiment. An experiment contains a number of blocks. The number of blocks in an experiment is a constant, 1000, which applies to all the experiments. Each block contains a number of trials. The number of trials can be 128, 256, 512, or 1024. Each trial is a judgment of the signal's presence on right or left interval. The judgment is either right or wrong. After we have gone though all the trials in a block, we can get a proportion of correct judgments and incorrect judgments. Thus we can derive two values. By equation (2.15) and (2.17), we obtained a value related to

$\Delta d'$. By equation (2.18) we obtained another value related to the estimated bias c. We consider the two values as two variables, because they depend on the exact test block from which they come. These two variables form a point in a Cartesian plane. After we go though all the test blocks in an experiment, we get 1000 points. Our objective is to examine the relationship between the two variables.

The judgement threshold is also a variable. It could be any value in the range of zero to the amplitude of signal. When we set it to zero, we assumed that we were simulating an unbiased observer. Otherwise, we just assumed we were simulating a biased observer. Since the noise array has mean equal to 0, and variance equal to 1, the SNR equals the signal amplitude by the following equation.

$$\text{SNR} = \text{signal}/\sigma \qquad (4.1)$$

Consider the 256 trials as typical experiments. In this kind of experiment, an experiment contains 1000 test blocks, and a test block contains 256 trials. In each block, a pair of statistical values can be obtained. After the whole experiment is done, a thousand pairs of corresponding values can be collected and depicted on a plot.

The resultant chart is exactly what we are interested in. Using different signal amplitude and various judgment threshold, a series of different but similar looking charts can be obtained.

Consider the following plot in Figure 4.1. This plot is a result of a 256-trials experiment. The signal amplitude is 2, so is the SNR. The observer judgment is set to 0. The observer compares this value with the difference between two stimulus from the two intervals for judgment. This process is designed to simulate an ideal observer with no bias.

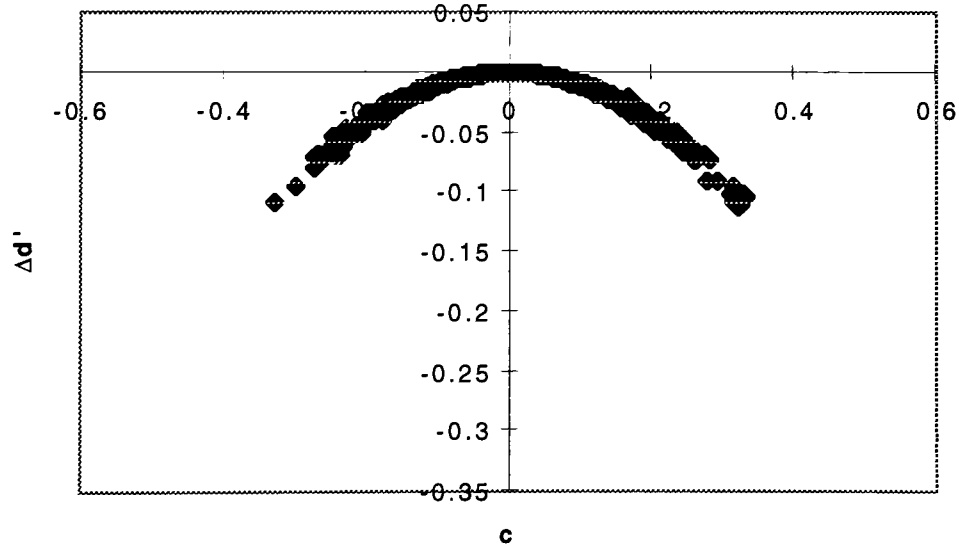


Figure 4.1. Plot of $\Delta d'$ versus c for experiment of 256 trials, SNR = 2, threshold = 0.

If there is a curve which can simulate the shape of these points, then it could be a parabola. This is just an intuitive sense.

Suppose it is a parabola, then this equation should be established, $\Delta d' = a \bullet c^2$.

In order to get the exact value of the coefficient a, the relationship of Δd' versus c^2 could be a big help. In this case, the coefficient is the slope of the line. So the following chart is derived.

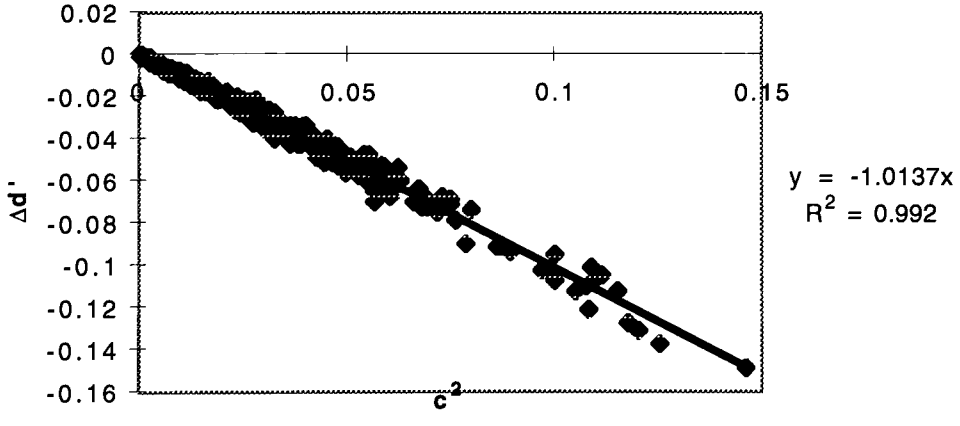


Figure 4.2. Plot of Δd' versus c^2 for experiment of 256 trials, SNR = 2, threshold = 0.

Microsoft Excel5.0 software is used in this thesis to depict plots and also do regression. As the plot shows, the slope is nearly -1, and the R^2 is quite good.

Take another experiment as an example. In this example , every experiment condition is the same as last one, except for the number of trials. 128 trials are chosen to be in a test block. The signal is 2 , and the threshold is 0.

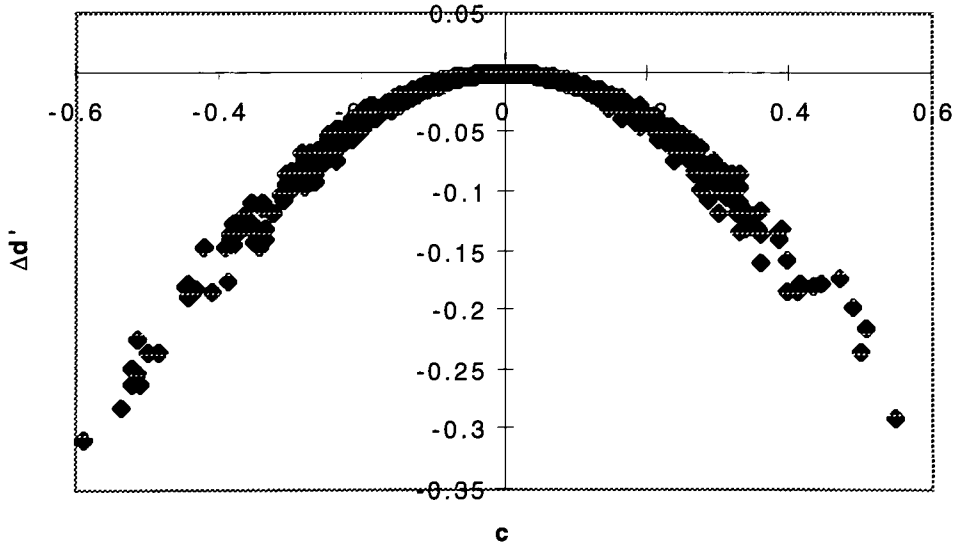


Figure 4.3. Plot of $\Delta d'$ versus c for experiment of 128 trials, SNR = 2, threshold = 0.

This also appears as a parabola. To figure out the coefficient a of the parabola, the relationship between Δd' and c^2 is plotted below

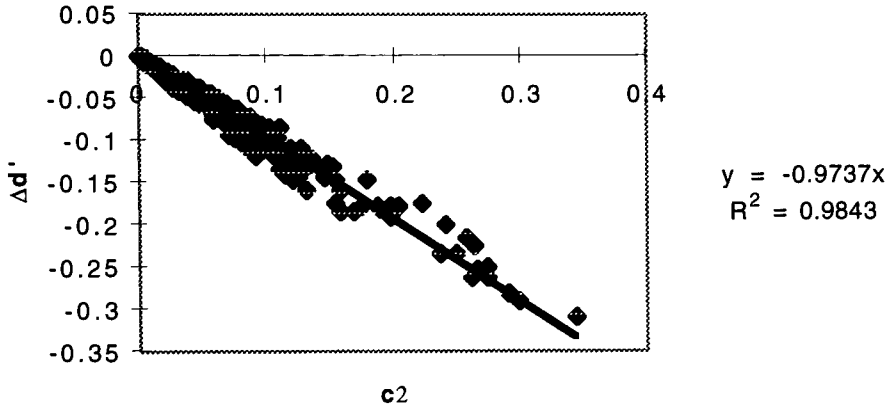


Figure 4.4. Plot of Δd' versus c^2 for experiment of 128 trials, SNR = 2, threshold = 0.

So this slope is also nearly -1. and the R^2 value is very good too. The reason why the points is a little bit more scattered, is that the fewer trials have been used.

Now, let us take the 512 trials experiment and 1024 trials experiment as instances, the signal amplitude and the judgment threshold are kept as 2 and 0 respectively.

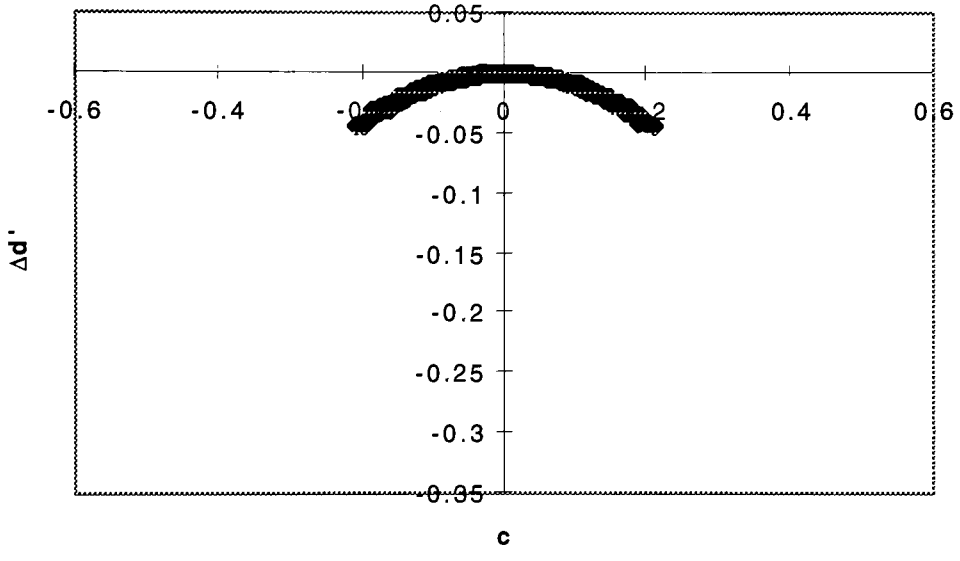


Figure 4.5. Plot of $\Delta d'$ versus c for experiment of 512 trials, SNR = 2, threshold = 0.

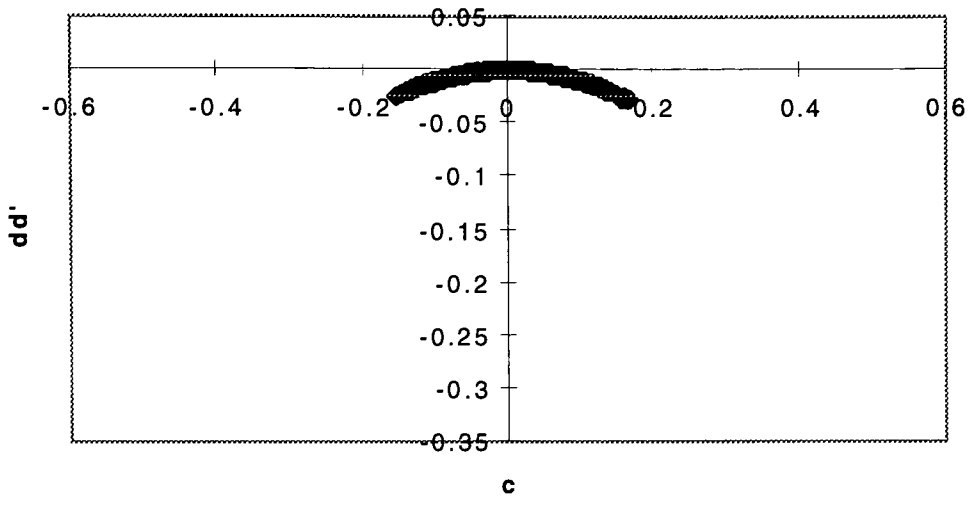


Figure 4.6. Plot of $\Delta d'$ versus c for experiment of 1024 trials, SNR = 2, threshold = 0.

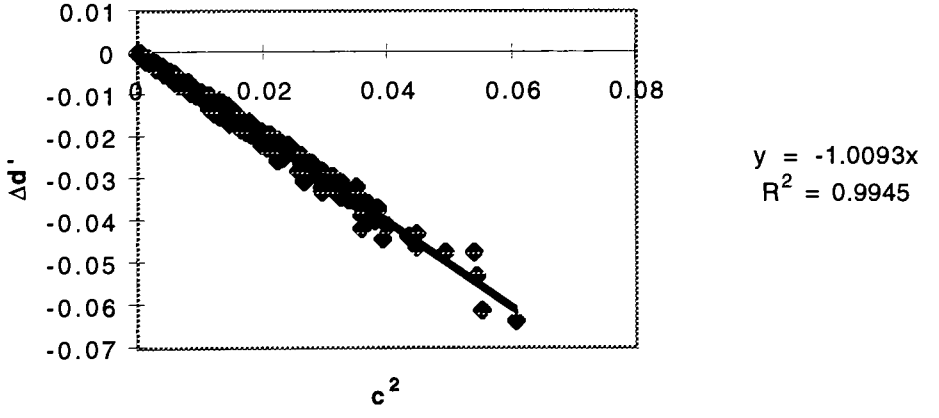


Figure 4.7. Plot of $\Delta d'$ versus c^2 for experiment of 512 trials, SNR = 2, threshold = 0.

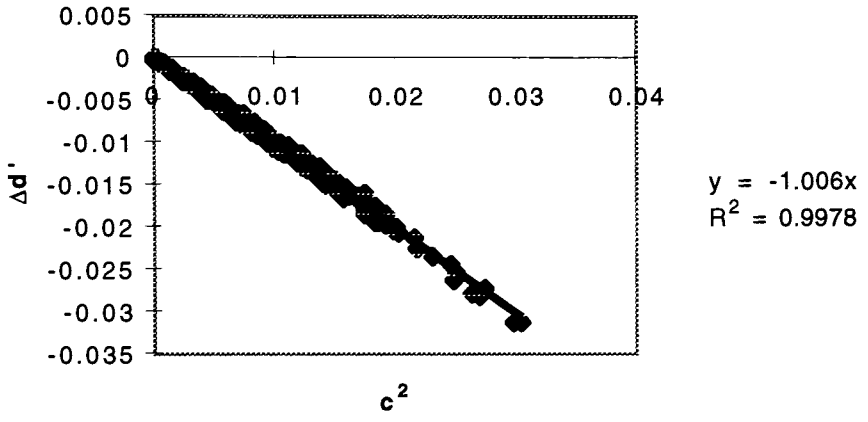


Figure 4.8. Plot of $\Delta d'$ versus c^2 for experiment of 1024 trials, SNR = 2, threshold = 0.

The same conclusion can be obtained, i.e. the coefficient is -1. And the more trials in an experiment, the better value for the R^2. But they do not make much difference since number of trials in a experiment is big enough for such a kind of experiment.

The above analysis is based on the condition that the signal and judgment threshold are unchanged. However if they are changed, what will happen?

Let us change the signal amplitude first. By changing the signal, we changed the signal to noise ratio in fact. In the following series of experiments, all the signal amplitudes are set to 1, so is the SNR. The judgment threshold is unchanged.

The following charts are obtained.

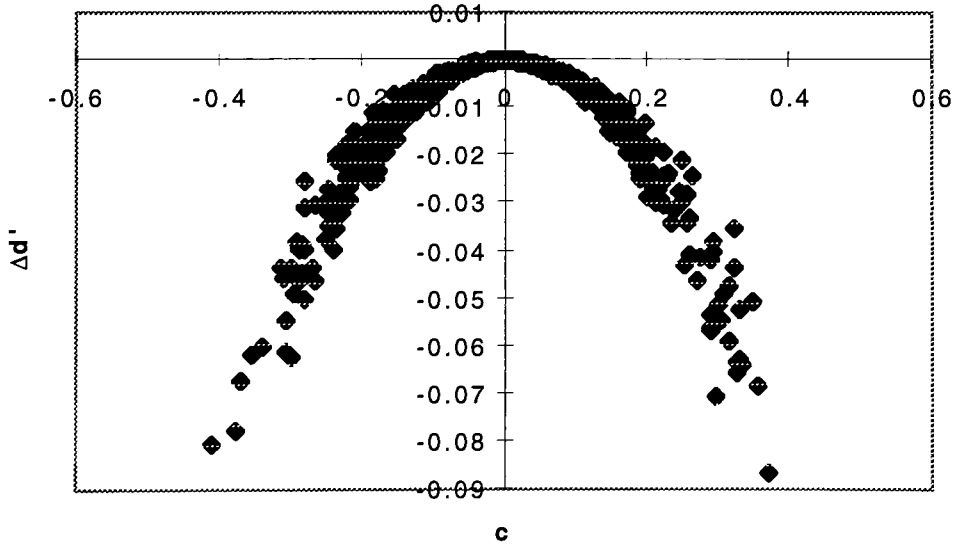


Figure 4.9. Plot of $\Delta d'$ versus c for experiment of 128 trials, SNR = 1, threshold = 0.

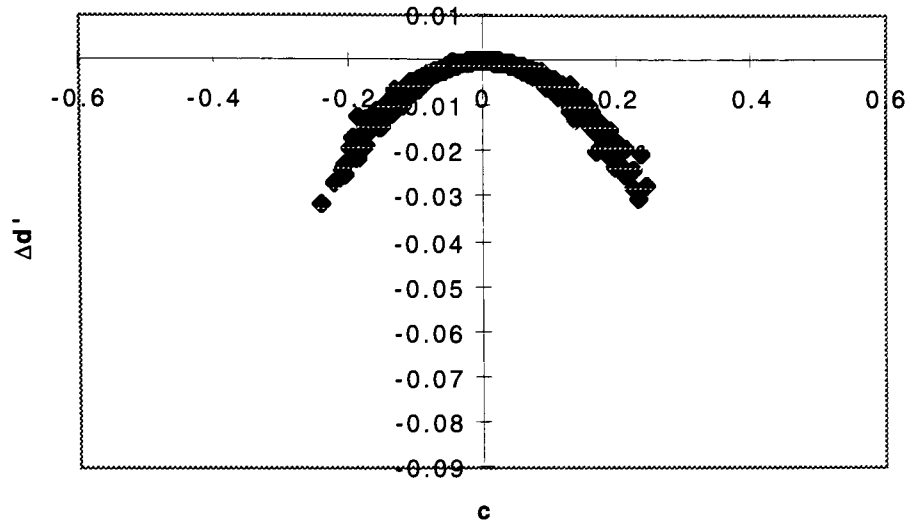


Figure 4.10. Plot of Δd' versus c for experiment of 256 trials, SNR = 1, threshold = 0.

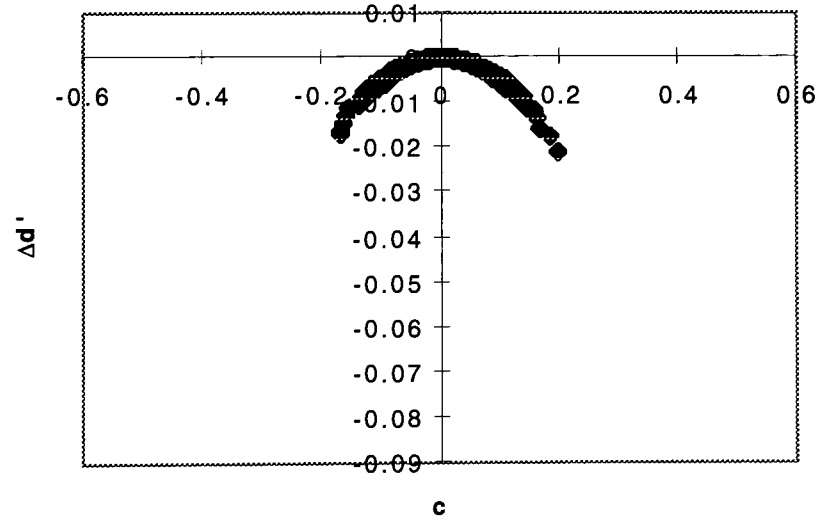


Figure 4.11. Plot of Δd' versus c for experiment of 512 trials, SNR =1, threshold = 0.

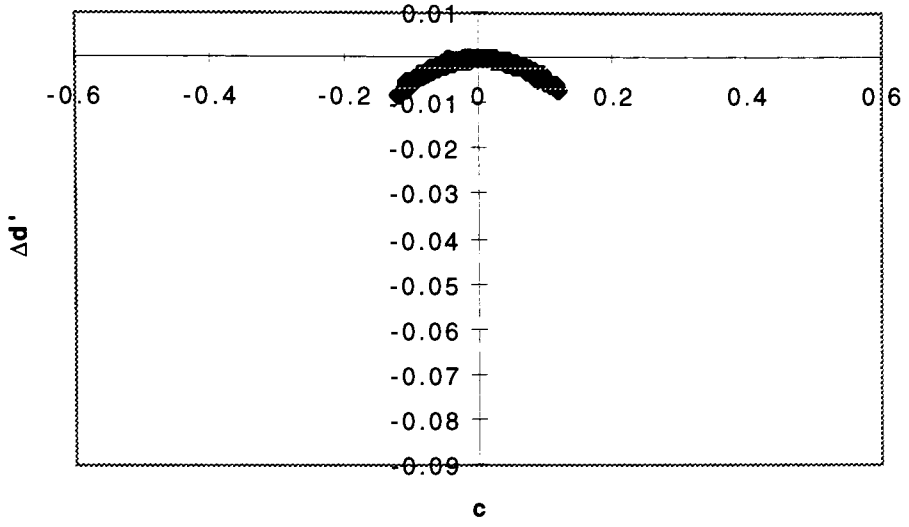


Figure 4.12. Plot of $\Delta d'$ versus c for experiment of 1024 trials, SNR = 1, threshold = 0.

and also we can get the plot of $\Delta d'$ versus c^2.

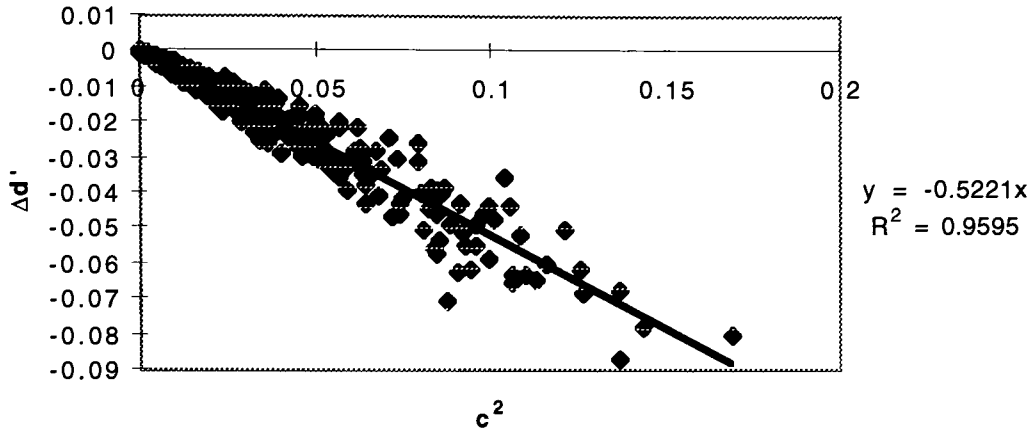


Figure 4.13. Plot of $\Delta d'$ versus c^2 for experiment of 128 trials, SNR = 1, threshold = 0.

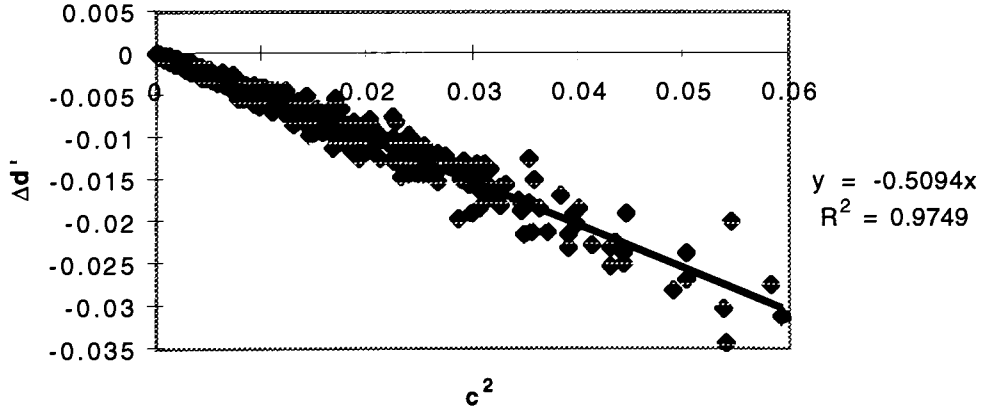


Figure 4.14. Plot of $\Delta d'$ versus c^2 for experiment of 256 trials, SNR = 1, threshold = 0.

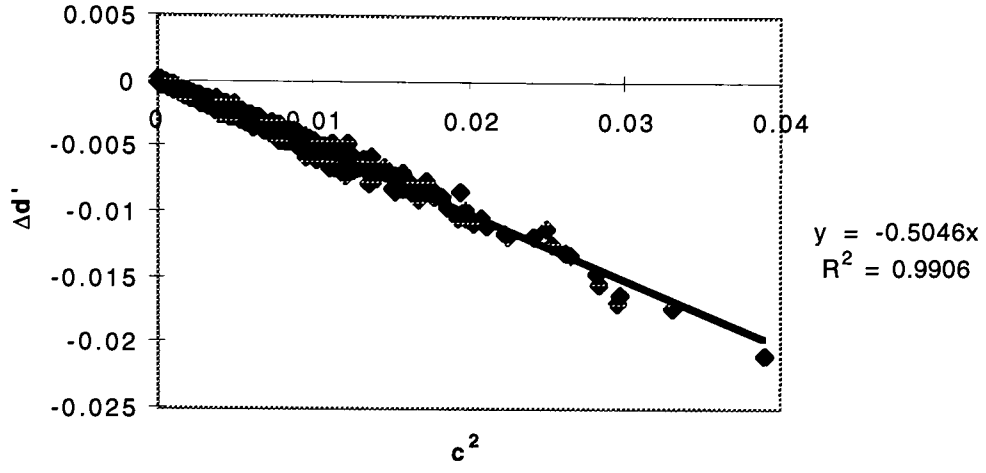


Figure 4.15. Plot of $\Delta d'$ versus c^2 for experiment of 512 trials, SNR = 1, threshold = 0.

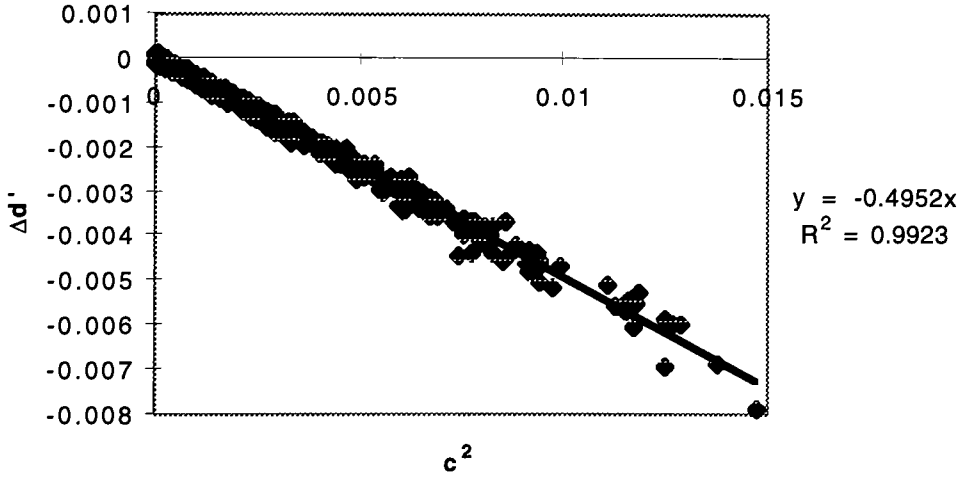


Figure 4.16. Plot of $\Delta d'$ versus c^2 for experiment of 1024 trials, SNR = 1, threshold = 0.

the conclusion is the coefficient is nearly -0.5, which is half of the value while the signal to noise ratio is 2.

The experiments mentioned above are designed to simulate an unbiased observer. The judgment threshold used by such an observer is always 0, which means the observer has no preference to either left side or right side in this 2AFC experiment. But the fact is that not all observers are unbiased. For those biased observers, what will be the relationship between $\Delta d'$ and the c? To get this relationship, first of all, a simulator of biased observer should be established. In our experiments, various biased observer simulators are obtained by changing the judgment threshold. For instance, if the threshold is set to 0.2 instead of 0, then only those values greater than 0.2 other than 0 are considered to be in right side, while all the values less than 0.2 are considered to be in left side. So such a kind of observer is a biased observer.

Taking 256 trials experiments for analysis, the signal amplitude was set to 2, hence the SNR is 2. The judgment threshold changes as 0, 0.2, 0.5, 1.0. A number of typical plots are shown below.

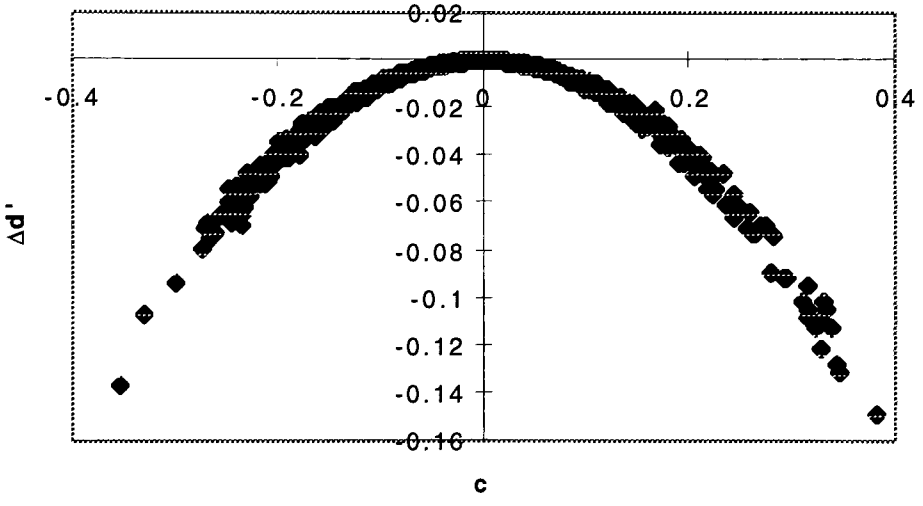


Figure 4.17. Plot of Δd' versus c for experiment of 256 trials, SNR = 2, threshold = 0.

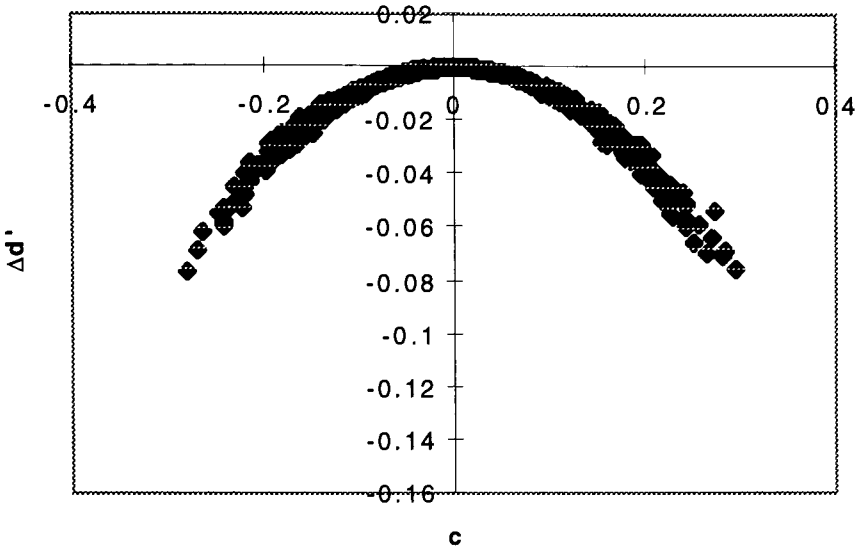


Figure 4.18. Plot of Δd' versus c for experiment of 256 trials, SNR = 2, threshold = 0.2.

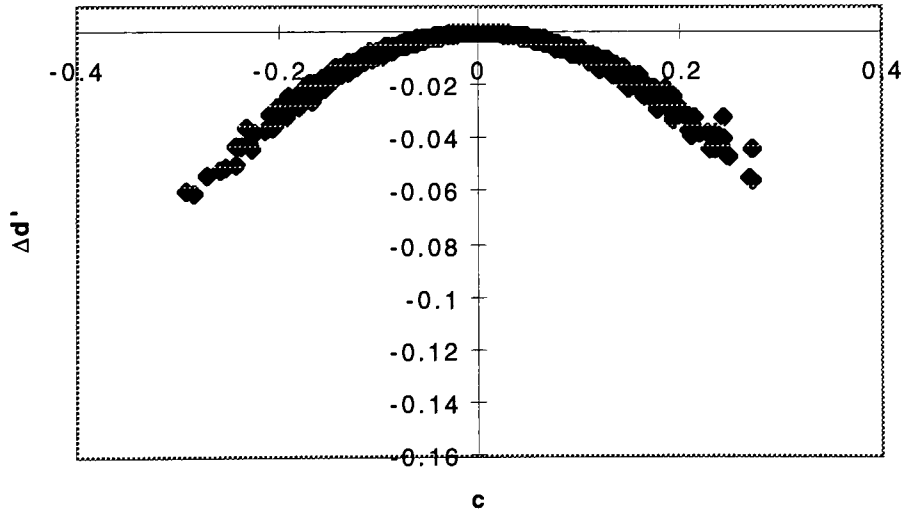


Figure 4.19. Plot of Δd' versus c for experiment of 256 trials, SNR = 2, threshold = 0.5.

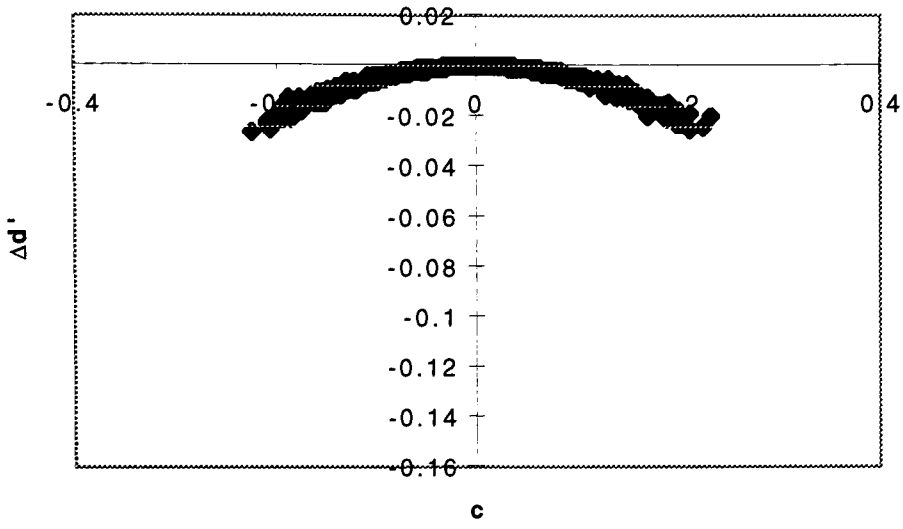


Figure 4.20. Plot of Δd' versus c for experiment of 256 trials, SNR = 2, threshold = 1.0.

Apparently, the arch becomes flatter as the judgment threshold increases. To know how much changes threshold cause, the following graph would be helpful. These charts provides the value of the coefficients.

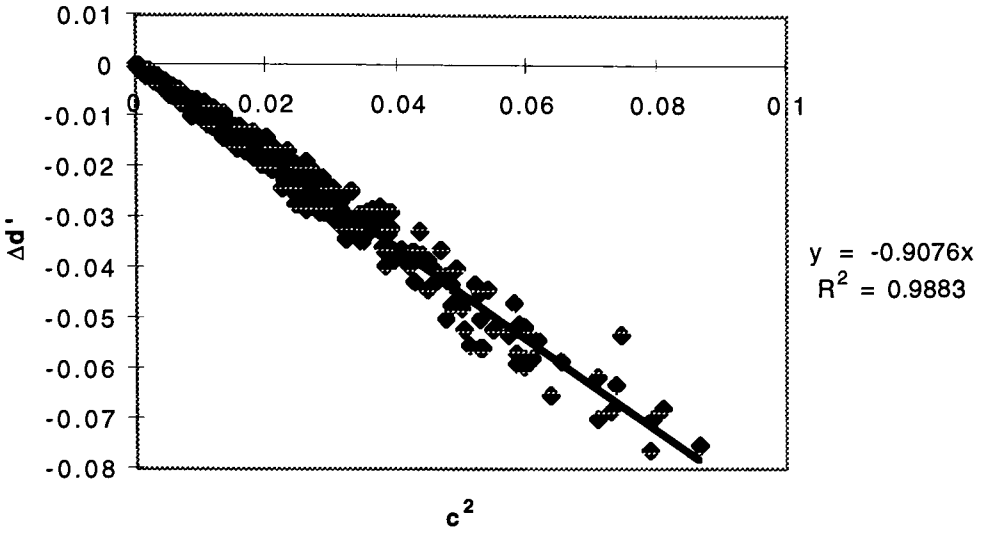


Figure 4.21. Plot of $\Delta d'$ versus c^2 for experiment of 256 trials, SNR = 2, threshold = 0.2.

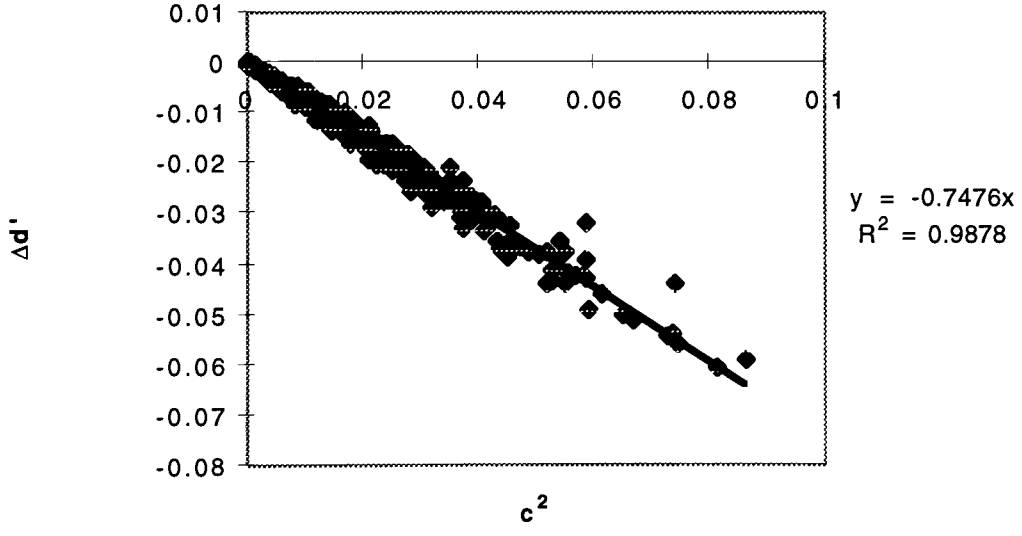


Figure 4.22. Plot of $\Delta d'$ versus c^2 for experiment of 256 trials, SNR = 2, threshold = 0.5.

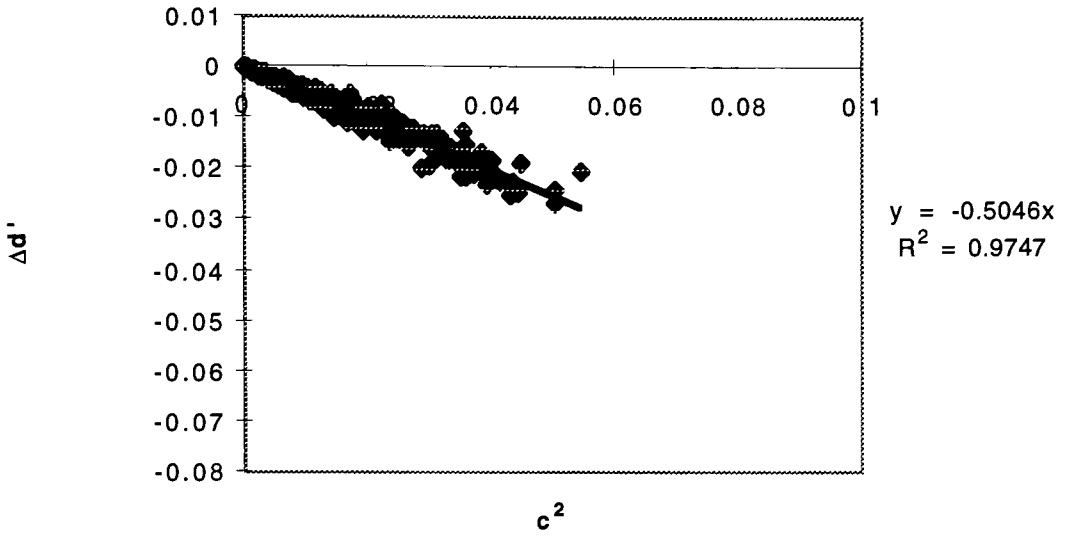


Figure 4.23. Plot of $\Delta d'$ versus c^2 for experiment of 256 trials, SNR = 2, threshold = 1.0.

The purpose of the experiments is to find the relationship between d' related parameters and the estimated bias c. So far, one can only see that the $\Delta d'$ and c^2 does have a kind of relation, like the curve of $\Delta d'$ versus the c^2 looks pretty much like a straight line. But the coefficient is not a constant. This key value keeps changing with the change of either signal amplitude or judgment threshold. If there exists a fixed coefficient which applies for all the cases, then the relationship between the $\Delta d'$ and c^2 is determined. After a lot of

experiments have been accomplished, we found out that the relationship we are searching for does exist, and we also determined its value.

On the way to find out the such a fixed relationship, one important step is to normalize the Δd'. The vertical variable, that used to be Δd', is now changed to the normalized value, Δd'/d'z. d'z is the unweighted detectability index calculated using unweighted average of z-scores. The horizontal variable remains unchanged. The following plots from the experiments supply more detailed information.

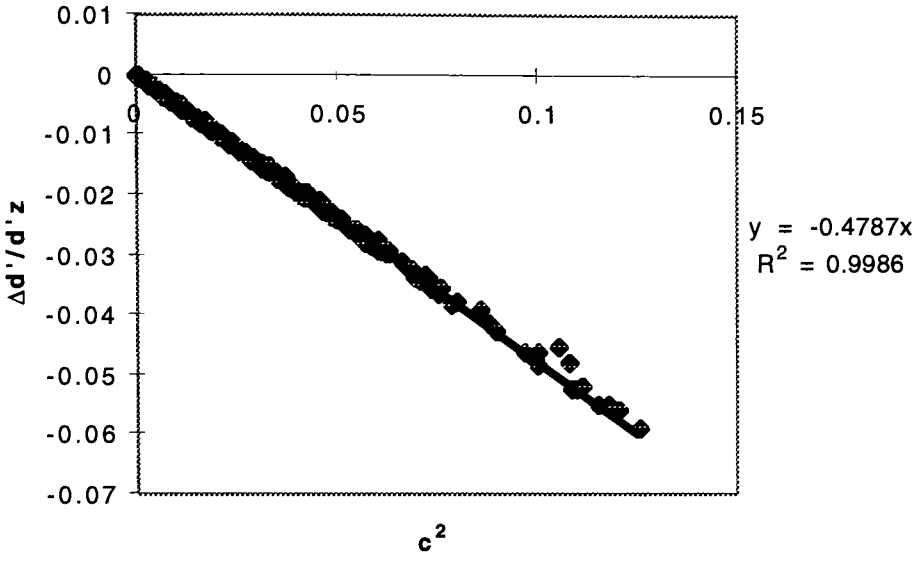


Figure 4.24. Plot of $\Delta d'/d'z$ versus c^2 for experiment of 256 trials, SNR = 2, threshold = 0.

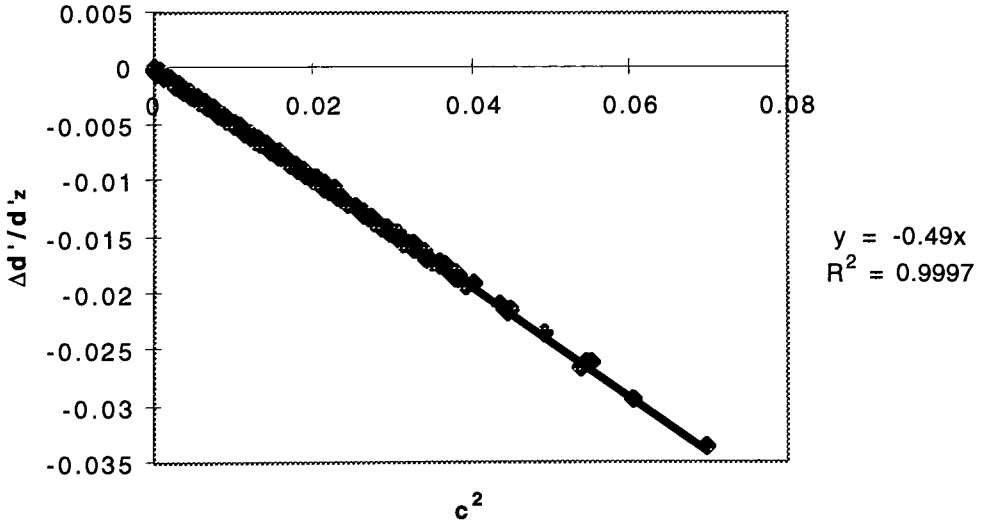


Figure 4.25. Plot of $\Delta d'/d'z$ versus c^2 for experiment of 512 trials, SNR = 2, threshold = 0.

Figure 4.24 and 4.25 show that the relationship between $\Delta d'/d'z$ and c^2 satisfy $\Delta d'/d'z = a \bullet c^2$, and the coefficient a equals nearly -0.5.

If the signal amplitude is changed, and everything else is held constant, the next two plots are obtained.

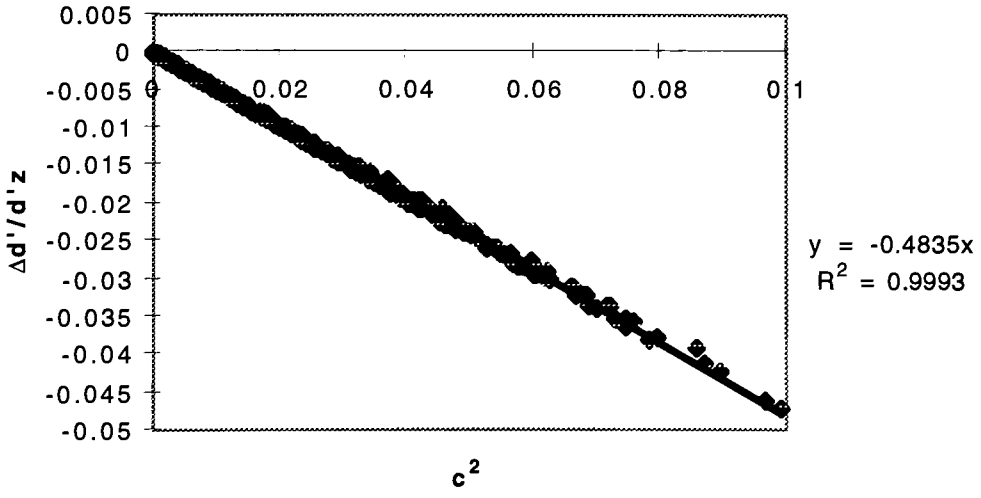


Figure 4.26. Plot of $\Delta d'/d'z$ versus c^2 for experiment of 256 trials, SNR = 1, threshold = 0

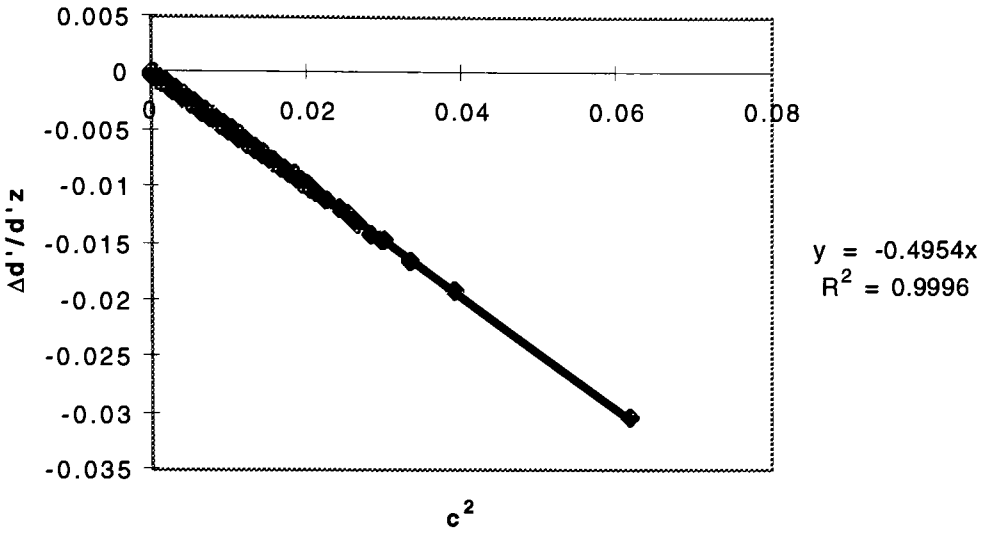


Figure 4.27. Plot of $\Delta d'/d'z$ versus c^2 for experiment of 512 trials, SNR = 1, threshold = 0.

The relationship of $\Delta d'/d'z = a \bullet c^2$ is still satisfied. and the coefficient a is the same, nearly -0.5.

In the experiment of simulating biased observers, the judgment threshold was changed while the other experimental conditions were constant. The results are shown below.

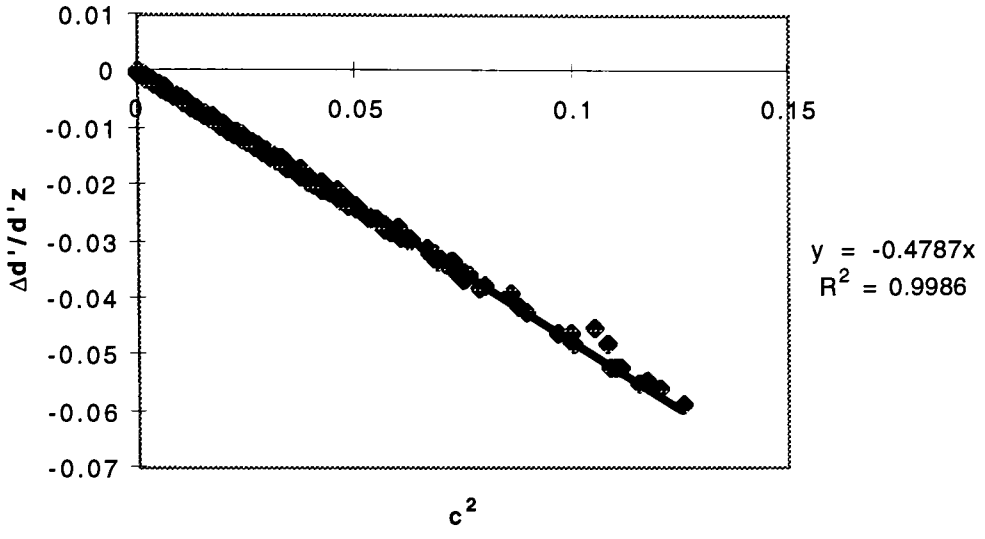


Figure 4.28. Plot of $\Delta d'/d'z$ versus c^2 for experiment of 256 trials, SNR = 2, threshold = 0.5.

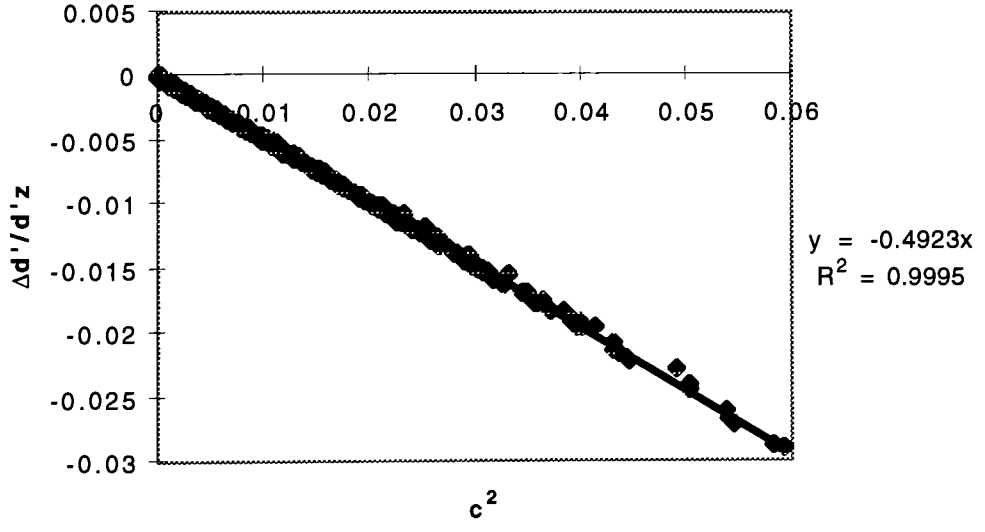


Figure 4.29. Plot of $\Delta d'/d'z$ versus c^2 for experiment of 256 trials, SNR = 2, threshold = 1.0

The equation $\Delta d'/d'z = a \bullet c^2$ fits the above results quite well. The coefficient a is also the same , -0.5.

So, now we can say that, for any case, no matter whether the observer is biased or unbiased, no matter its signal amplitude is the same or not, the relationship

$$\Delta d'/d'z = -0.5\ c^2$$

is true if the threshold is less than the signal amplitude. The larger the difference between the signal amplitude and the judgment threshold, the better the data fit the above equation.

The above analysis is based on the results obtained from Monte Carlo simulation methods. The observers are not real observers, but are computer simulated ones. In the real world, a number of experiments have been done also. The following experimental results demonstrate the equation $\Delta d'/d'z = -0.5\ c^2$ applies in the real cases as well.

5. Observer Performance

In the real world, the observer performance experiment is designed as follows.

The visual stimuli comes from a CRT display with two square areas side-by-side each occupied by background noise. This noise is generated as independent Gaussian noise. The mean and the variance of the Gaussian noise are both controllable. The signal in our experiment is a disk signal centered on one of the two square areas. The amplitude of the signal can be adjusted in different experiments. The signal is randomly assigned to one of the two square areas. In one experiment 256 stimuli are included in a given condition, which means the mean, variance of the Gaussian noise and the signal amplitude can not be changed during an experiment.

The equipment in the human observer experiment included a DEC MicroVax computer, a Peritek VCU-Q display board, and a Ramtek GM-714 monitor. The display board was configured to give

240 × 320 pixels. The two side-by-side square areas were configured to give 32 × 32 pixels. The signals were sharp-edged and had a radius of 4 pixels, which subtended to a visual angle of around 1 degree.

The observers viewed the image stimuli in a dimly illuminated room. The observer can move around freely in the dark room while viewing the stimuli. He or she is forced to judge which side contains the signal. From the observer's point of view, there will be one of four results, correct right, false right, correct left, or false left. The definition of correct right is that the signal is on the right side and the observer chooses the right. The false right is that the signal is on the left side and the observer chooses the right. The correct left is the signal on the left and the observer chooses the left. And the last one, the false left, is the signal on the right and the observer chooses the left.

Ten observers have done a large number of experiments. The data I used to plot was existing data previously collected in the lab. I

selected three plots to show the relationship between the detectability index and the estimated bias. The rest of the data demonstrate similar results. The data obtained from the real observers matched the computer simulation data. Plots are shown in following figures.

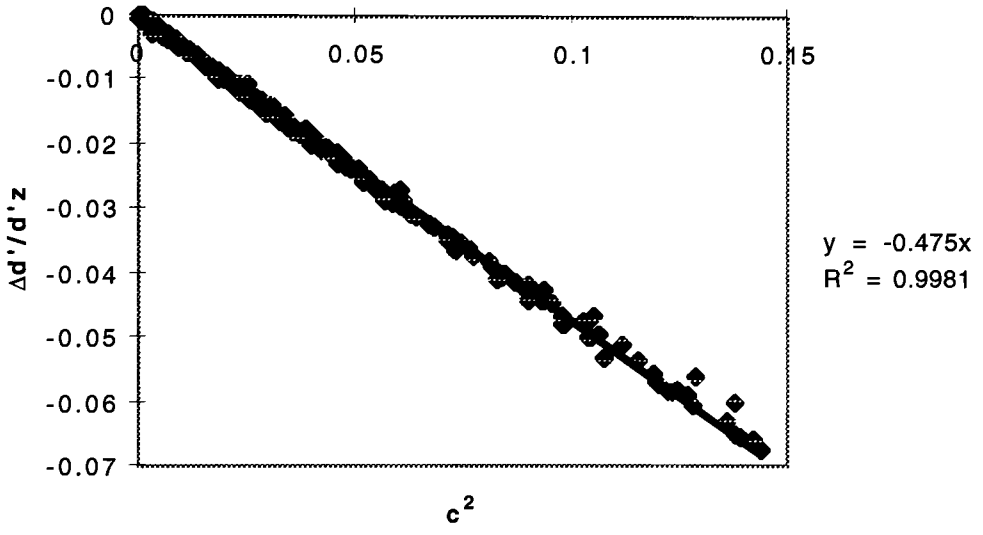


Figure 5.1. Plot of $\Delta d'/d'$ versus c^2 for experiment of observer AB after he had done 399 test blocks. There were 256 test trials in each test block (test case f020).

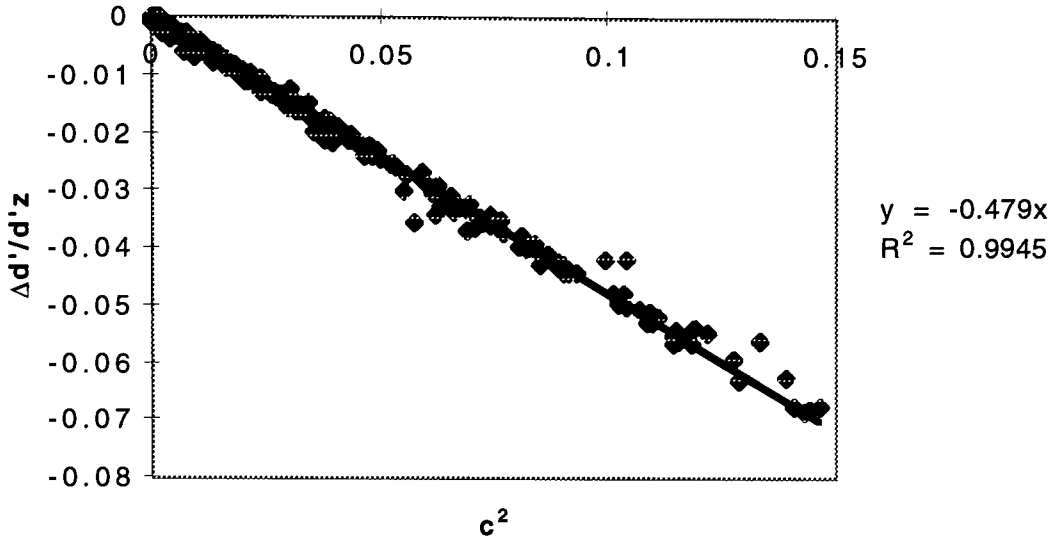


Figure 5.2. Plot of $\Delta d'/d'$ versus c^2 for experiment of observer AB after he had done 435 test blocks. There were 256 test trials in each test block (test case f060).

As the figure shows, the relationship between the normalized difference, $\Delta d'/d'$, and the estimated observer bias, c, is approximately parabolic; $\Delta d'/d' = a \bullet c^2$. The figures also show that the coefficient a is approximately -0.5. The human observer results also shows the same relationship.

Figure 5.1 and 5.2 show one observer's performance in different test cases. Test case f020 and f060 represent different

variances of Gaussian noise applied in those test cases. The SNRs were selected to give d' in the range from 1.5 to 2.5.

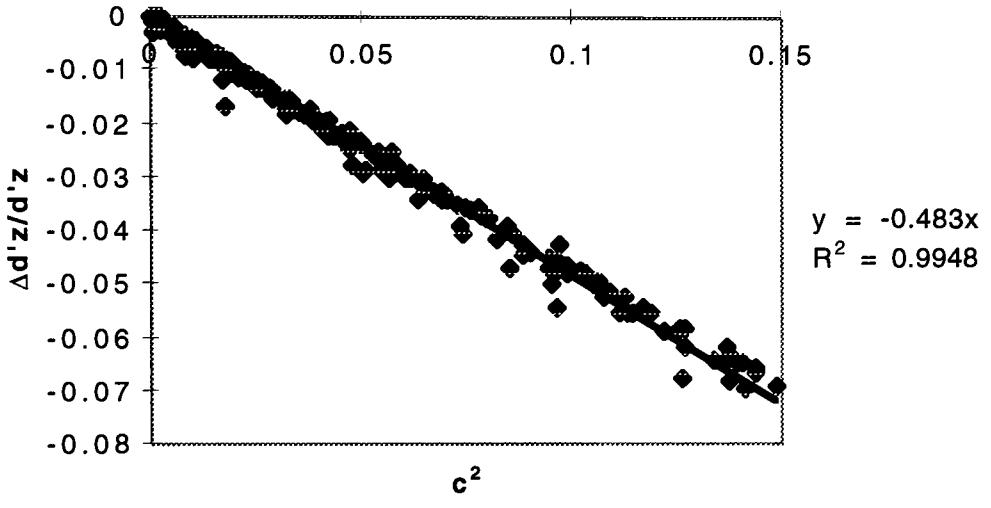


Figure 5.3. Plot of $\Delta d'/d'$ versus c^2 for experiment of observer YQ after she had done 498 test blocks. There were 256 test trials in each test block (test case f060).

Figure 5.2 and 5.3 show two observer's performance in one test case. Test case f060 represent variance of Gaussian noise applied in this test case. The SNRs were selected to give d' in the range from 1.5 to 2.5.

Obviously, the results obtained from the Monte Carlo simulations and from the human observers match up to one another very well.

6. Closed Form Calculation of $\Delta d'$ versus bias

This calculation will be done using Taylor's series expansions about the point corresponding to z_m on the P(z) function. With reference to Figure 6.1 shown below, let the proportion correct and z-scores of points 1 and 2 be (P_1, z_1) and (P_2, z_2) respectively. Then $z_m=(z_1+z_2)/2$ and $c=(z_2-z_1)/2$.

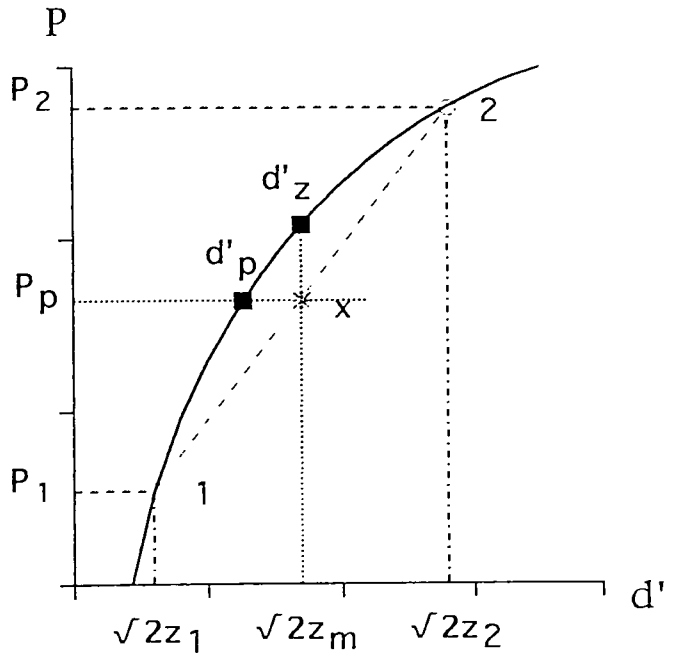


Figure 6.1. Plot of probability P versus detectability index d'.

We will use Taylor's series and the notation P', P'', P''', P'''' to indicate derivatives of P evaluated at z_m. We get:

$$P_1 = P(z_1) = P(z_m - c)$$

$$\approx P(z_m) - cP' + c_2P''/2 - c_3P'''/6 + c_4P''''/24 - \ldots \quad (6.1)$$

and

$$P_2 = P(z_2) = P(z_m + c)$$

$$\approx P(z_m) + cP'' + c_2P''/2 + c_3P'''/6 + c_4P''''/24 \quad \ldots \quad (6.2)$$

The mean value of P evaluated in the p-domain is given by

$$P_P = (P_2 + P_1)/2 \approx P(z_m) + c_2P''/2 + c_4P''''/24 + \ldots \quad (6.3)$$

and hence

$$P_P - P(z_m) \approx c_2P''/2 + c_4P''''/24 + \ldots \quad (6.4)$$

We can use another Taylor's series to evaluate the difference, $P_P - P(z_m)$. Letting $z_p = z(P_P)$ and $\Delta z = z_p - z_m$, we get the approximation

$$P_P = P(z_m + \Delta z) \approx P(z_m) + \Delta zP' + \Delta z^2P''/2 + \Delta z^4P''''/24 + \ldots \quad (6.5)$$

which can be used to obtain:

$$\Delta zP' \approx (c^2 - \Delta z^2)P''/2 + (c^4 - \Delta z^4)P''''/24 + \ldots \quad (6.6)$$

Then using the relationship between P and z from equation (2.1) we find:

$$P' = \exp[-(z_m)^2/2] \; /\sqrt{2\pi} \tag{6.7}$$

$$P'' = -z_m P' \tag{6.8}$$

$$P'''' = 3zP' - (z_m)^3 P' \tag{6.9}$$

Replace P'' and P'''' in equation (6.6):

$$\Delta z P' \approx (c^2 - \Delta z^2)(-z_m P')/2 + (c^4 - \Delta z^4)(3z_m P' - (z_m)^3 P')/24 \tag{6.10}$$

$$\Delta z \approx (-c^2 z_m)/2 + (\Delta z^2 z_m)/2 \; - c^4 (z_m)^3/24 + 3z_m c^4/24 + \ldots \tag{6.11}$$

The lowest order approximation from equation (6.6) can now be evaluated.

$$\Delta z = (z_p - z_m) \approx \; -z_m c^2/2 \tag{6.12}$$

Recalling that $d' = \sqrt{2}\, z$, we get:

$$\text{and} \qquad (d'_P - d'_z)/d'_z \approx -c^2/2 \tag{6.13}$$

We can use this low order result in (6.9) to approximate the Δz^2 term in (6.6) and get the second order result:

$$(z_P - z_m) \approx -z_m c^2/2 + (z_m)\, 3c^4/12 \qquad (6.14)$$

and

$$(d'_P - d'_Z)/d'_Z \approx -c^2[1 - 3c^2/12 - (cd'_Z)^2/12] / 2 \qquad (6.15)$$

$$\approx -c^2[1 - (c^2/12)(3 - d'_Z{}^2)] / 2 \qquad (6.16)$$

The term in square brackets is the approximation to the coefficient, a, discussed in the main text. A similar recursive approach could be used to determine higher order approximations to the coefficient. Since the probabilities of correct responses over both left and right intervals are around 0.9 in our test cases, the value of d'_Z can be approximated to 2 [1]. Therefore the term $(3 - d'_Z{}^2)$ can be approximated to 1, and the term in square bracket can be approximated to $[1 + (c^2/12)]$. For small c, the above equation reduces to $\Delta d'/ d'_Z \approx -c^2/2$.

7. Conclusions

In summary, Monte Carlo simulation can be used in 2AFC experiments to achieve a very successful result. The Gaussian noise generator we adopted here is quite satisfactory, and plays a key role in each experiment, and ensures the accuracy of the data upon which we draw our conclusions.

The Monte Carlo simulator is satisfactory in its performance. It simulated both unbiased and biased observers, and accomplished various decision making tasks successfully.

After different results from different experimental cases were analyzed, it was found that $\Delta d'$ does have relationship to c^2. The relationship can be expressed as

$$\Delta d' = a \bullet c^2$$

where the coefficient a varies with signal amplitude and observer's judgment threshold. The value of a is always negative, and the maximum value is 0.

Depending on the change caused by signal amplitude or judgment threshold, the value of the coefficient a changes in opposite directions. If the signal amplitude increases, a will decrease. If the observer's judgment threshold increases, a will increase. This is reasonable, since the closer to zero a is, the worse the performance of the observer. When the signal increases, the signal to noise ratio will increase. The bigger SNR will improve the observer's judgment performance, and therefore makes a farther away from 0, i.e. a becomes smaller. On the other hand, when the judgment threshold increases, the observer will not be able to distinguish the signal between the 2 images unless the difference between the two images is larger than the threshold. This will result in the observer not making a correct response when the signal is drowned out by a strong noise background. This will decrease the observer's judgment

performance, and therefore makes a closer to 0, i.e. a becomes larger.

However, if the $\Delta d'$ is normalized, then the relationship between $\Delta d' / d'$ and c^2 is determined. The relationship can be expressed as:

$$\Delta d'/d' \approx a \bullet c^2$$

where a is a constant, and equal to -0.5. This equation defines the relationship for all the 2AFC cases. Despite the changes of signal amplitude and judgment threshold, this relationship is always true for small estimated bias.

We also derived a closed form to calculate the relationship using Taylor's series expansion approach. The approximate relationship between the detectability difference $\Delta d'$ and the estimated bias c is :

$$(d'_P - d'_Z)/d'_Z \approx -c^2[1 - 3c^2/12 - (c\ d'_Z)^2/12] / 2$$

For small bias c, this equation reduces to: $\Delta d'/d' \approx -0.5 \bullet c^2$.

In the real world, the same relationship exists for the human observer's performance results. This result strongly supports the Monte Carlo simulation method and the Monte Carlo simulator designed and implemented in this thesis.

8. References

1. Neil A. Macmillan and C. Douglas Creelman, *"Detection Theory: A User's Guide"* , Cambridge University Press, NY, (1991)

2. William T. Vetterling, Saul A. Teukolsky, William H. Press and Brian P. Flannery, *"Numerical Recipes Example Book (C)"* , Cambridge University Press, NY, (1988). Reprinted (1988), (1989)

3. Arther E. Burgess, *"Comparison of ROC and Forced Choice Observer Performance Measurement Methods"* , Med. Phys. 22, (1995)

4. Edward R. Dougherty, *"Probability and Statistics Engineering, Computing, and Physical Science "* , Prentice Hall, Englewood Cliffs, New Jersey, (1990)

5. David M. Green and John A. Swets, *"Signal Detection Theory and Psychophysics "*, John Wiley and Sons, Inc. , New York, (1966)

6. I. M. Sobol, *"The Monte Carlo Method "*, Translated and adapted from the second Russian edition by Robert Messer, John Stone, and Peter Fortini, The University of Chicago Press, Chicago, (1974)

7. Reuven Y. Rubinstein, *"Simulation and the Monte Carlo Method "*, John Wiley and Sons, New York, (1981)

8. George G. Lowry, *"Markov chains and Monte Carlo Calculations in polymer science"*, Marcel Dekker, Inc. , New York, (1970)

9. P.B. Elliott, *"Signal detection and recognition by human observers "*, John Wiley, New York, (1964)

10. R. G. Swensson and P. F. Judy, *"Measuring observer efficiency and consistency for visual discrimination in noise backrounds "*, J. Exp. Psych: Human Perception and Performance (in press)

11. John A. Swets, *"Signal detection and recognition by human observers"*, John Wiley and Sons, New York, (1964)

12. Athanasios Papoulis, *"Probability, random variables, and stochastic processes"*, McGraw-Hill, New York, (1984)

9. Symbol Definitions

N = the total number of trials in each test block.

N_l = the number of trials in which the left side contains the signal.

N_r = the number of trials in which the right side contains the signal.

N_{l_c} = the number of trials in which the left side contains the signal and the observer made a correct judgment

N_{l_e} = the number of trials in which the left side contains the signal and the observer made an incorrect judgment

N_{r_c} = the number of trials in which the right side contains the signal and the observer made a correct judgment

N_{r_e} = the number of trials in which the right side contains the signal and the observer made an incorrect judgment

P = the probability

$P_l = N_l/N$, the probability of signal on left side

$P_r = N_r/N$, the probability of signal on right side

$P_{l\_e} = N_{l\_e}/N$, the probability of incorrect response while signal on left side

$P_{r\_e} = N_{r\_e}/N$, the probability of incorrect response while signal on right side

$P_u = (P_l + P_r)/2$, unweighted average of probability

$P_w = (N_{l\_c} + N_{r\_c})/ (N_l + N_r)$, weighted average of probability

z-score = the inverse of normal distribution function.

$d' = (1/\sqrt{2})[z(H) - z(F)]$, detectability index

$c = -0.5[z(H) + z(F)]$, estimated bias

$d'_{pu} = (1/\sqrt{2})z(Pu)$, unweighted detectability index, d' , calculated using unweighted probability

$d'_{zu} = (1/\sqrt{2})[z(P_l) + z(P_r)]/2$, unweighted detectability index, d' , calculated using unweighted average of z-score

d'_{pw} = weighted detectability index, d' , calculated using weighted average of probability

d'_{zw} = weighted detectability index, d' , calculated using weighted average of z-score

$\Delta d' = d'_{zu} - d'_{pu}$, difference of detectability index

10. Appendix

The C pograms used to do Monte Carlo simulation experiments.

10.1 Appendix A

The C pograms for Monte Carlo simulation.

```
/*****************************************************************************
File name : thesis_main.c
Author:     Dalei Huang

Description: This file includes the main function for Monte Carlo simulation
*****************************************************************************/

#include <stdio.h>
#include <math.h>


/*****************************************************************************
              Function calls
*****************************************************************************/

int size(void);
float  * signalgenerator(int n, float signal_value);
float * gaussnoisegenerator(int n, int seed);
void   addnoise (float * p1,float * p2, int n);
float cal_mean(double * p,int n);
float cal_var(double * p,int n,float m );
float prob_of_smallpart(float * signalandnoise, float thresholdsub,int size);
float zscore (double p,double mean,double var);


/*****************************************************************************
               Main Function
*****************************************************************************/

int main()
{
        /*************************************************
                definition
        *************************************************/
        float *p_noise1,*p_noise2, *p_noise_assign;
        float threshold;
        int trials_of_block;
        int number_of_blocks;
        int noiseseed1,noiseseed2,noiseseed3;
        float sig_amplitude;
        float sig_at_right, sig_at_left;
        float correct_right, correct_left;
        float p_cor_l, p_cor_r, p_w, d_prime_pw,d_prime_zu,d_prime_zw,delta_d_p;
        float final_value ;
        float z_value_left,z_value_right;
        float d_z,c,c_square;
        int i,j,k=0;

        int trials[4]={128,256,512,1024};
        int tr;

        FILE  *fp1[4];


        /**** Following are output files  ************/
        /*--- CHANGE here for different file name ---*/
        fp1[0] = fopen("NewData/128_1_0.gong","w");
        fp1[1] = fopen("NewData/256_1_0.gong","w");
        fp1[2] = fopen("NewData/512_1_0.gong","w");
        fp1[3] = fopen("NewData/1024_1_0.gong","w");
```

```
number_of_blocks = 1000;


sig_amplitude = 1;     /*--- CHAANGE here for different SIGNAL ---*/

threshold = 0;         /*--- CHANGE here for different THRESHOLD -*/

k = 0 ;                /*--- index number for file ------*/


for (tr=0;tr<4;tr++){

   trials_of_block  = trials[tr];        /*-- trials can be 128,256,512,1024 ---*/
                                         /*-- depends on the array trials[] value */

    for(i=1;i<=number_of_blocks;i++){

        /*** Select noise seeds ******/
        noiseseed1 =  -509999 + 5397*i;
        noiseseed2 =  -507777 + 5397*i;
        noiseseed3 =  -505759 + 5397*i;

        /*** Generate Gaussian noise ************************************/
        p_noise1 = (float*)gaussnoisegenerator(trials_of_block, noiseseed1);
        p_noise2 = (float*)gaussnoisegenerator(trials_of_block, noiseseed2);
        p_noise_assign = (float*)gaussnoisegenerator(trials_of_block,noiseseed3);


        /*** initialization ***/
        sig_at_right=0;
        sig_at_left=0;
        correct_right=0;
        correct_left=0;


        /*** decide the signal on which side ***/
        for(j=0;j<trials_of_block;j++){

                if(p_noise_assign[j] <0) {
                        sig_at_left++;
                        if(p_noise1[j]+sig_amplitude - p_noise2[j] > threshold)
                        correct_left++;
                }else{
                        sig_at_right++;
                        if(p_noise2[j]+sig_amplitude - p_noise1[j] > threshold)
                        correct_right++;
                }
        }


        /*** Calculate the correct probability of each side ***/
        p_cor_l = correct_left/sig_at_left;
        p_cor_r = correct_right/sig_at_right;

        /*** Calculate correct probability of the whole experiment ****/
        p_w = (correct_left + correct_right)/
                (sig_at_left + sig_at_right);
```

```
   /**** calculate the d'pw ******************************/
   d_prime_pw = sqrt(2.0)*(float)zscore((double)p_w,0.0,1.0);


   /**** calculate z-value    ******************************/
   z_value_left = zscore((double)correct_left/sig_at_left,0.0,1.0);
   z_value_right = zscore((double)correct_right/sig_at_right,0.0,1.0);


   /**** calculate d'zu and d'zw ***************************/
   d_prime_zu = sqrt(2) * (z_value_left + z_value_right)/2;

   d_prime_zw = sqrt(2) * (sig_at_left*z_value_left +
                       sig_at_right *z_value_right)/trials_of_block;

   /**** calculate delta d'p **********************/
   delta_d_p = d_prime_pw - d_prime_zw ;

   /**** calculate the normalized delta d'p ******/
   final_value = delta_d_p/d_prime_zu ;

   /**** calculate the bias c **************/
   c =  (z_value_left - z_value_right)/2 ;

   /**** calculate the c_square ************/
   c_square = pow( (z_value_left - z_value_right)/2 , 2.0);



   /*** Depending on what relationship we want, choose different output data ***/
   /*for c versus delta_d'     */
   fprintf(fp1[k],"%f         %f\n\n", c,delta_d_p);

   /*for c_square versus delta_d'/d'z    */
   /*fprintf(fp1[k],"%f         %f\n\n", c_square,final_value);
   */


   free (p_noise1);
   free (p_noise2);
   free (p_noise_assign);

   }

 k = k+1;

} /*end of tr<4 loop */


    /**** free pointers *****/
    for(k=0;k<4;k++){
        fclose(fp1[k]);
    }

return (0);
}
```

```
/*********************************************************************
File Name: cal_zscore.c
Auther:    Dalei Huang

Description: This file provides a function to calculate
             the z_score value, given the probability.
*********************************************************************/

#include "../BGS933/gauss.c"
#define width 0.0001


/*********************************************************************
Function prototype
*********************************************************************/

float zscore (double p,double mean,double var);


/*********************************************************************
function name: float zscore (double p,double mean,double var)
Description:   calculate the z_score value.
*********************************************************************/

float zscore (double p,double mean,double var)

{       float x;
        double fn_value,unit_area,sum;
        double d;
        double maxsum = 0;

        sum = 0;

        if (p>=0.99997) x=5.0;
        else{

            for (x=(-5);sum <=p;x=x+width){
                fn_value = (float)gauss((double)x,(double)mean,(double)var);
                unit_area = fn_value * width ;
                sum = sum + unit_area ;
            }
        }

        d = sqrt(2) * x ;

        return (x);
}
```

```
/*****************************************************************
File Name: thesis.c
Auther·    Dalei Huang

Description: This file implements the following functions,
             which are called by the thesis main function.

             int size(void);
             float  * signalgenerator(int n, float signal_value);
             float * gaussnoisegenerator(int n, int seed);
             void   addnoise (float * pl,float * p2, int n);
             float cal_mean(float * p,int n);
             float cal_var(float * p,int n,float m );
             float prob_of_smallpart(float * signalandnoise,
                                     float thresholdsub,int size);
             float zscore (double p,double mean,double var);
*****************************************************************/


#include <stdio.h>
#include <math.h>
#include "../BGS933/isa.h"
#include " ./BGS933/ran1.c"
#include "../BGS933/gauss_ran1.c"


/*****************************************************************
Fuction prototypes
*****************************************************************/

int size(void);
float  * signalgenerator(int n, float signal_value);
float * gaussnoisegenerator(int n, int seed);
void   addnoise (float * p1,float * p2, int n);
float cal_mean(float * p,int n);
float cal_var(float * p,int n,float m );
float prob_of_smallpart(float * signalandnoise, float thresholdsub,int size);
float zscore (double p,double mean,double var);



/*****************************************************************
Fuction:       int size()
Description:   prompts the user to enter a size
*****************************************************************/

int size (void)
{
        int n;
        printf("input trial numbers (sampling numbers of signal):\n");
        scanf("%d",&n);

        return (n);
}


/*****************************************************************
Fuction:       float  * signalgenerator(int n, float signal_value)
Description:   generates a array with size equal to n,
                              and value equal to signal_value
```

```
*****************************************************************/

float  * signalgenerator(int n, float signal_value)
{

        float  * pointer;
        int i;

        pointer =(float  *) calloc(n,sizeof(float));

        for(i=0;i<n;i++){
                pointer[i] = signal_value;
        }

        return (pointer);
}



/*****************************************************************
Fuction:        float * gaussnoisegenerator(int n, int seed)
Description:    generates a array of gaussian noise values.
                        the size of array is n.
*****************************************************************/

float * gaussnoisegenerator(int n, int seed)
{
        float * pointer;
        int i;

        printf ("Enter gaussnoisegenerator\n");

        pointer =(float *) calloc(n,sizeof(float));

        for (i=0;i<n;i++){
                pointer[i] = gauss_ran1(&seed);

        }
        printf ("Exit gaussnoisegenerator\n");
        return(pointer);
}



/*****************************************************************
Fuction:        void   addnoise (float * p1,float * p2, int n)
Description:    add two arrays of gaussian noise.
                        the size of array is n.
*****************************************************************/

void   addnoise (float * p1,float * p2, int n)
{
        int i;
        for(i=0;i<n;i++){
                p1[i] = p1[i] + p2[i];
        }
}


/*****************************************************************
```

```
Fuction:        float cal_mean(float * p,int n)
Description:    calculate the mean of noise array.
*****************************************************************/

float cal_mean(float * p,int n)
{
        int i;
        float sum =0.0;
        float mean;

        for(i=0;i<n;i++){
            /*printf("noise   =  %f, i = %d\n",p[i], i );*/
            sum = sum + p[i];
        }
        mean = sum/n;

        return (mean );
}



/*****************************************************************
Fuction:        float cal_var(float * p,int n,float m )
Description:    calculate the variance of noise array.
*****************************************************************/

float cal_var(float * p,int n,float m )
{
        int i;
        float sum =0,var;

        for (i=0;i<n ;i++){
                sum = sum + (p[i] -m)* (p[i]-m);
        }

        var = sum/(n);

        return (var );
}


/*****************************************************************
Fuction:        float prob_of_smallpart(float * signalandnoise,
                                        float thresholdsub,int size)
Description:    calculate the obability of noise less than threshold.
*****************************************************************/

float prob_of_smallpart(float * signalandnoise, float thresholdsub,int size)
{
        float checknumber=0.0;
        float prob;
        int i;

        for(i=0;i<size;i++){
                if (signalandnoise[i] <= thresholdsub){
                        checknumber=checknumber +1;
                }

        }
```

```
        prob = checknumber/size;
        return (prob);
}
```

10.2 Appendix B

The C pograms for calculation of histograms.

```
/* read data from a file
   calculate the histogram
                mean and var
*/
#include <stdio.h>
#include <math.h>


int size(void);
float  * signalgenerator(int n, float signal_value);
float * gaussnoisegenerator(int n, int seed);
void   addnoise (float * pl,float * p2, int n);
float cal_mean(float * p,int n);
float cal_var(float * p,int n,float m );
float prob_of_smallpart(float * signalandnoise, float thresholdsub,int size);
float zscore (double p,double mean,double var);

int main()
{
        float  p[100000];
        float step;
        FILE * fl;
        int i;
        int trials_of_block;
        int  * hist;
        int bin_num;
        float  Start,
            End;
        int ii;
        float variable;
        float mean,  var ;

        trials_of_block = 100000 ;

        Start = -5 ;
        End = 5;
        step = 0.2 ;

        bin_num = (End - Start) / step ;

        hist = (int *) malloc ( bin_num, sizeof(int));

        fl = fopen("noise.dat","r");


        for (i =0 ; i< trials_of_block; i++){
                fscanf(fl,"%f",&p[i]);
                /*printf("%f\n", p[i]);*/
        }
                printf (" i = %d \n",i);

         for ( ii = 0 ; ii <= bin_num; ii++){
                hist[ii] = 0;
         }

        for ( ii = 0 ; ii < bin_num; ii++){
                variable = Start + ii * step ;
                /*printf ("variable is %f\n",variable);*/

                for (i = 0; i< trials_of_block; i++){
```

```
                if( ((p[i]- variable) < step) && ((variable - p[i]) < step) )
                        hist[ii] ++;
                }
                printf ("%f     %d\n",variable, hist[ii] );
        }

        mean = (float) cal_mean( p, trials_of_block);
        var = (float) cal_var( p,trials_of_block ,mean );

        printf (" mean = %f\n", mean);
        printf ("var = %f\n", var);

                printf ("exit main      \n");

        free (hist);

        return (0);
}
```

10.3 Appendix C

The C pograms for generating Gaussian noise.

```
/*****************************************************************************
File name   noise_gen.c
Author:     Dalei Huang


Description: This file includes the main function for noise generator
*****************************************************************************/

#include <stdio.h>
#include <math.h>

int size(void);
float  * signalgenerator(int n, float signal_value);
float * gaussnoisegenerator(int n, int seed);
void   addnoise (float * p1,float * p2, int n);
float cal_mean(double * p,int n);
float cal_var(double * p,int n,float m );
float prob_of_smallpart(float * signalandnoise, float thresholdsub,int size);
float zscore (double p,double mean,double var);

int main()
{
        float *p_noise1,*p_noise2, *p_noise_assign;
        float threshold;
        int trials_of_block;
        int number_of_blocks;
        int noiseseed1,noiseseed2,noiseseed3;
        float sig_amplitude;
        float sig_at_right, sig_at_left;
        float correct_right, correct_left;
        float p_cor_l, p_cor_r, p_w, d_prime_pw,d_prime_zu,d_prime_zw,delta_d_p;
        float final_value ;
        float z_value_left,z_value_right;
        float d_z,c;
        int i,j,k=0;
        float mean, var;

        trials_of_block  = 100000;



                noiseseed1 =  -509999 ;

                p_noise1 = (float*)gaussnoisegenerator(trials_of_block, noiseseed1);


        for(j=0; j <trials_of_block; j++)
                        printf("%f\n",p_noise1[j]);

        free (p_noise1);


        return (0);
}
```

```
/*
** isa.h
**
** -- header file for using gauss noise generator
**
*/

#ifndef _ISA


#define UNSIGNED        0
#define SIGNED          1

/* prototypes for functions */
float ranl( int * );
float gauss_ranl( int * );
int   *calc_histogram( float *, int, int, int );

#define _ISA

#endif /* _ISA */
```

```
/* subroutine ran1_nrc.c

   ran1.c function from "numerical recipes in c" ref[2]
   use ran_1ink.com to 1ink subroutine with main program

*/

#define M1 259200
#define IA1 7141
#define IC1 54773
#define RM1 (1.0 / M1)
#define M2 134456
#define IA2 8121
#define IC2 28411
#define RM2 (1.0 / M2)
#define M3 243000
#define IA3 4561
#define IC3 51349

float ran1( int *idum )
{

   static long ix1, ix2, ix3;
   static float r[98];
   float temp;
   static int iff = 0;
   int j;

   if ( *idum < 0 || iff == 0 ) {

      iff = 1;
      ix1 = ( IC1 - (*idum) ) % M1;
      ix1 = ( IA1 * ix1 + IC1 ) % M1;
      ix2 = ix1 % M2;
      ix1 = ( IA1 * ix1 + IC1 ) % M1;
      ix3 = ix1 % M3;

      for ( j = 1; j <= 97; j++ ) {
          ix1 = ( IA1 * ix1 + IC1 ) % M1;
          ix2 = ( IA2 * ix2 + IC2 ) % M2;
          r[j] = ( ix1 + ix2 * RM2 ) * RM1;
      }

      *idum = 1;

   }

   ix1 = ( IA1 * ix1 + IC1 ) % M1;
   ix2 = ( IA2 * ix2 + IC2 ) % M2;
   ix3 = ( IA3 * ix3 + IC3 ) % M3;
```

```
    j = 1 + ( (97 * ix3) / M3 );

    if ( j > 97 || j < 1 ) {
       printf( "RAN1: This cannot happen." );
       exit( 1 );
    }

    temp = r[j];
    r[j] = ( ix1 + ix2 * RM2 ) * RM1;
    return temp;

}
```